SDLC

Sujata Batra

Sujata Batra - sujatabatra@hotmail.com

- IT Trainer Since 2000
- More than 50+ Corporate Clients

Sujata Batra

Definition

The Software Development Life Cycle (SDLC) model is

- An approach to have a linear sequence of steps to develop a system or software product
- To execute the process from start to finish without revisiting any previous step
- One of the oldest systems development models and is still the most commonly used

SDLC

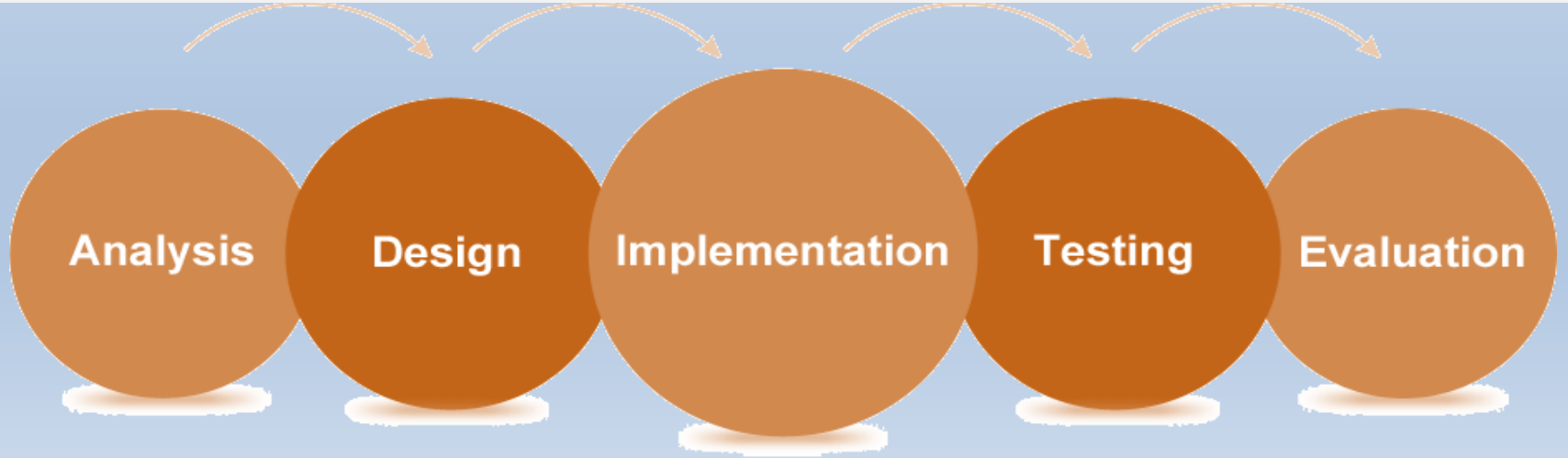
The systems development life cycle (**SDLC**) is a term used in:

Systems Engineering

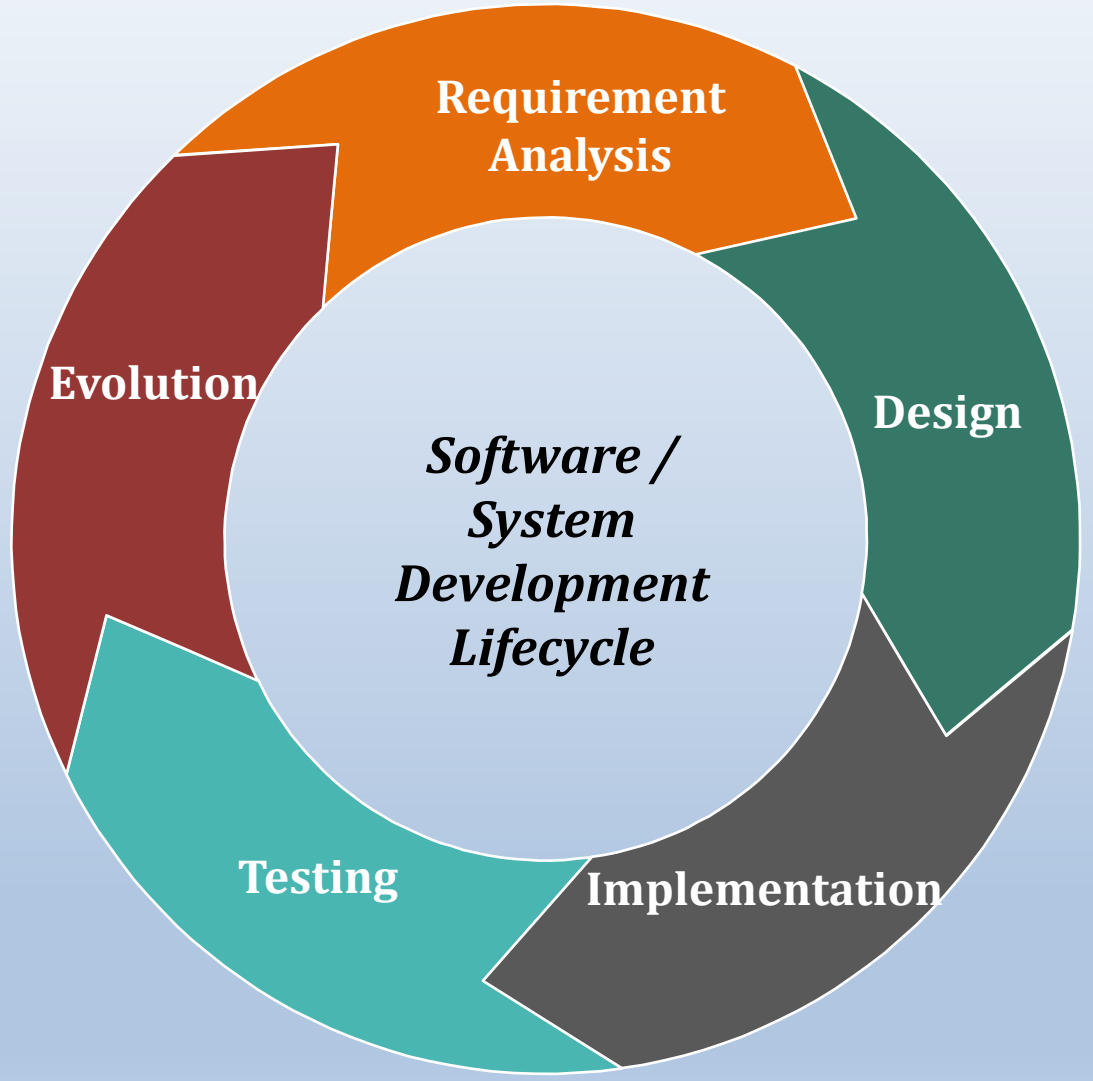
Information Systems

Software Engineering

Also called application development life-cycle.

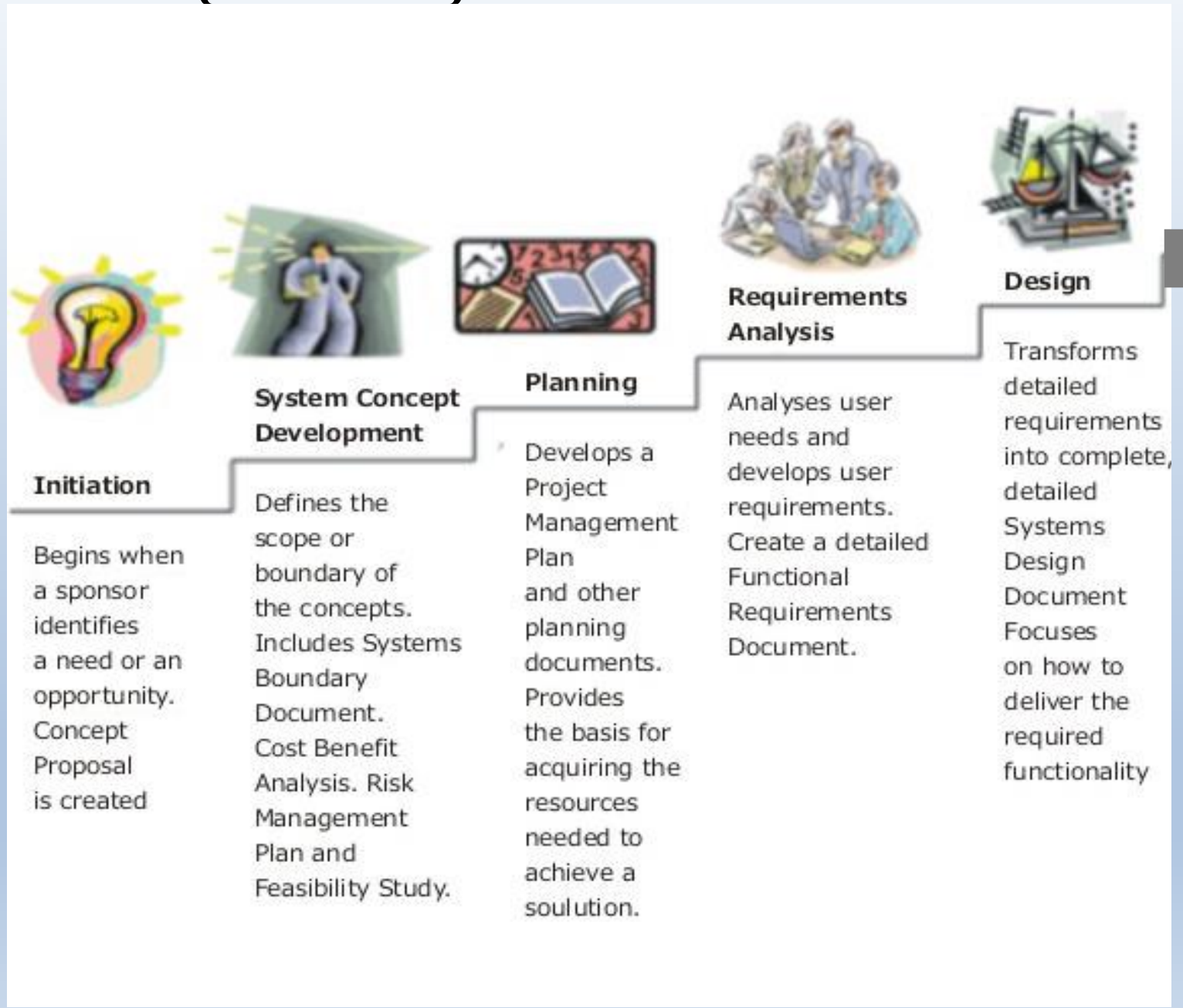


SDLC

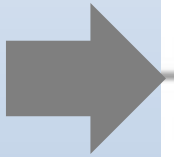


SDLC is the motherboard for all kinds of project developments!

SDLC Phases (Part 1 of 2)



SDLC Phases (Part 2 of 2)



Development

Converts a design into a complete information system Includes acquiring and installing systems environment; creating and testing databases preparing test case procedures; preparing test files, coding, compiling, refining programs; performing test readiness review and procurement activities.

Integration and Test

Demonstrates that developed system conforms to requirements as specified in the Functional Requirements Document. Conducted by Quality Assurance staff and users. Produces Test Analysis Reports.

Implementation

Includes implementation preparation, implementation of the system into a production environment, and resolution of problems identified in the Integration and Test Phases

Operations & Maintenance

Describes tasks to operate and maintain information systems in a production environment. includes Post-Implementation and In-Process Reviews.

Disposition

Describes end-of-system activities, emphasis is given to proper preparation of data.

Project Initiation Phase

This is the 1st phase in the Project Life Cycle, as it involves starting up a new project.

A project is started by defining its :

- Objectives
- Scope
- Purpose
- Deliverables

Also in this phase we hire the project team, setup the Project Office and review the project, to gain approval to begin the next phase.

The purpose of the Initiation Phase is to start the project.

8

Concepts Development Phase

The Concept Development Phase may begin after the approval of the completion of the Initiation project status review, and the approval to proceed to the Concept Development Phase.

The focus of the phase is two-fold:
1) Evaluate feasibility of alternatives and
2) Clearly define and approve project scope, including the system, all deliverables, and all required activities.

Planning

Project Planning – Determines the project's goals and results in a high-level view of the potential project.

Proper comprehensive project planning is essential to a successful IT project, and incomplete project planning and analysis are frequently root causes of project failure.

The purpose of the Planning Phase is to plan all project processes and activities required to ensure project success and to create a comprehensive set of plans, known as the Project Management Plan (PMP), to manage the project from this phase until project termination.

Requirement Analysis

The Requirements Analysis Phase begins when the previous phase objectives have been achieved.

Documentation related to user requirements from the Concept Development Phase and the Planning Phase shall be used as the basis for further user needs analysis and the development of detailed requirements.

The purpose of the Requirements Analysis Phase is to transform the needs and high-level requirements specified in earlier phases into unambiguous (measurable and testable), traceable, complete, consistent, and stakeholder-approved requirements.

Design

During the Design Phase, the system is designed to satisfy the requirements identified in the previous phases.

The requirements identified in the Requirements Analysis Phase are transformed into a System Design Document that accurately describes the design of the system and that can be used as an input to system development in the next phase.

The purpose of the Design Phase is to transform the requirements into complete and detailed system design specifications. Once the design is approved, the Development Team begins the Development Phase.

Development

The Development Phase features a key step in the project: system construction.

The previous phases lay the foundation for system development; the following phases ensure that the product functions as required.

To complete the Development Phase successfully, two elements are required:

1) 1) A complete set of design specifications
2) 2) Proper processes, standards, and tools.

The purpose of the Development Phase is to convert the system design prototyped in the Design Phase into a working information system that addresses all documented system requirements. At the end of this phase, the working system will enter the Test Phase.

Testing

The Test Phase focuses on an empirical investigation in which the results describe the quality of the system: testing cannot confirm a system functions properly under all conditions but can establish that it fails under specific conditions.

In the Test Phase, testing of the system proves that the system meets all requirements, including those for performance and security.

The purpose of the Test Phase is to guarantee that the system successfully built and tested in the Development Phase meets all requirements and design parameters. After being tested and accepted, the system moves to the Implementation Phase.

Implementation

The Implementation Phase has one key activity:
Deploying the new system in its target environment. Supporting actions include training end-users and preparing to turn the system over to maintenance personnel.

The purpose of the Implementation Phase is to deploy and enable operations of the new information system in the production environment.

Operation and Maintenance

During the Operations and Maintenance Phase, the information system's availability and performance in executing the work for which it was designed is maintained.

System operations continue until the system's termination date, when the next phase, Disposition, begins.

The purpose of the Operations and Maintenance Phase is to ensure the information system is fully functional and performs optimally until the system reaches its end of life.

Disposition

The Disposition Phase is the end of an information system's life cycle. The information system is formally retired according to organizational needs, laws and regulations, and the Disposition Plan.

The disposition activities ensure that the information system is terminated in an orderly manner and that vital information about the system is preserved according to applicable records management regulations and policies for future access.

The decision to proceed with the Disposition Phase is based on recommendations and approvals from an In-Process Review during the Operations and Maintenance Phase.

The purpose of the Disposition Phase is to shut down the operational information system in a controlled manner.

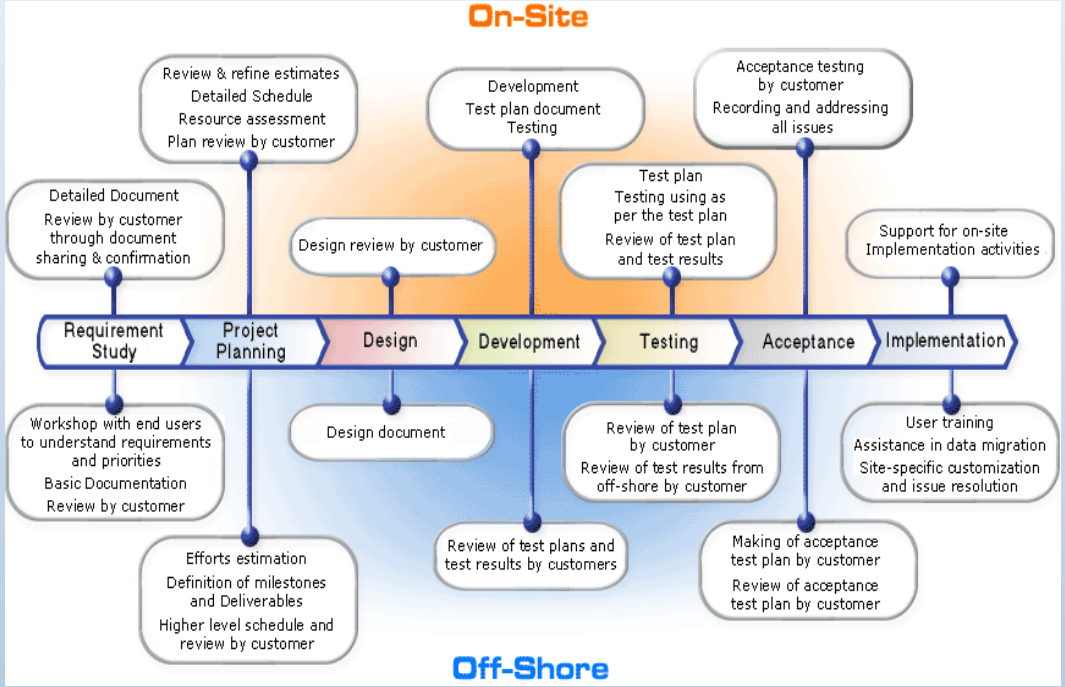
Why SDLC ?

We need to follow SDLC

- To execute projects with proven frame work
- To define and focus roles and responsibilities
- To enforce planning and control
- To have consistency among deliverables
- To increase productivity by executing the project in systematic manner
- To reduce the rework effort during project execution

Types of SDLC

- Waterfall Model
- Prototyping Model
- Incremental Model
- Spiral Model
- V – Model



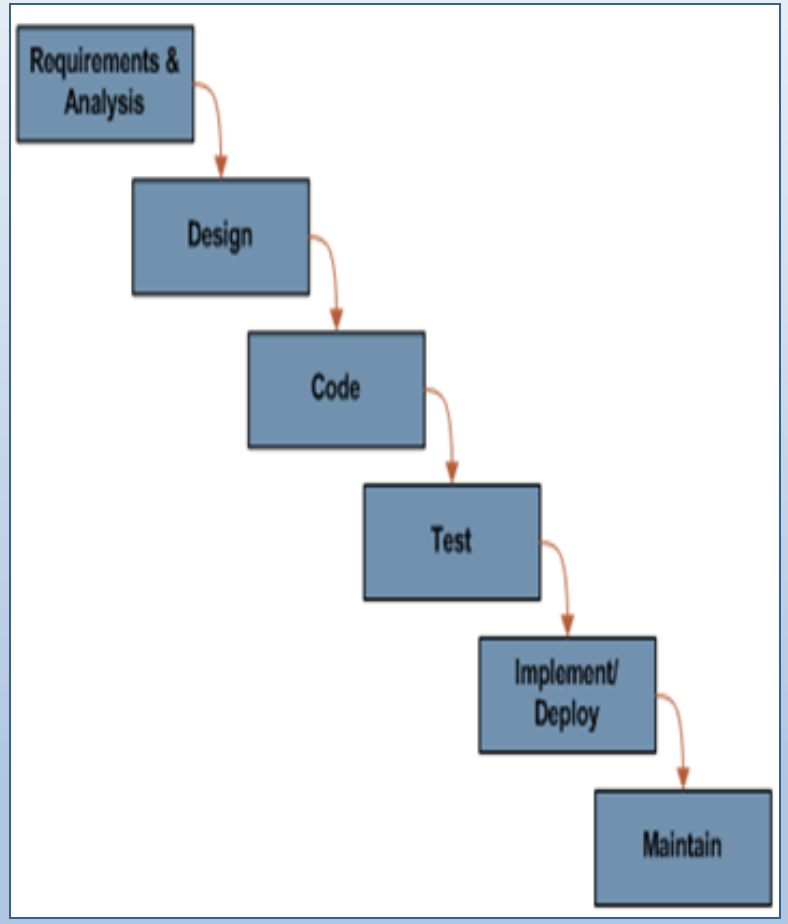
- Rapid Application Development Model (RAD)

Waterfall Model

Waterfall model is the base Model of SDLC

Main features are:

- Whole process of software development is divided into separate phases
- Derives from its name, giving cascading effect from one phase to another phase
- Each phase has well defined starting and ending point with identifiable deliveries to the next phase
- Most commonly used model



Waterfall Model Advantages

1. The most commonly used model because it is easy to use and understand.
2. No phase is considered to be complete until it is documented and verified
3. Provides means for making structured and stable development process, fostering the creation of high quality deliverables
4. Milestones are well – defined and understood

Waterfall Model Disadvantages

1. Requirements must be fully defined at the beginning itself.
2. It is difficult to get early feedback either on requirements feasibility or implementation approach
3. A working version of the software will not be available until late in the cycle
4. The customer and developer interaction is less during the development of the product

Prototyping Model

Prototyping is the process of quickly putting together a working model.

This model

- Provides proof of concept
- Gives users an idea of what the final system looks like
- Increases the system development speed
- Helps to identify any problems with earlier design
- Enables users to give quicker feedback on the approach
- Is Cost effective
- Is not the final product and suitable especially for the analysis phase

Prototype Model Advantages

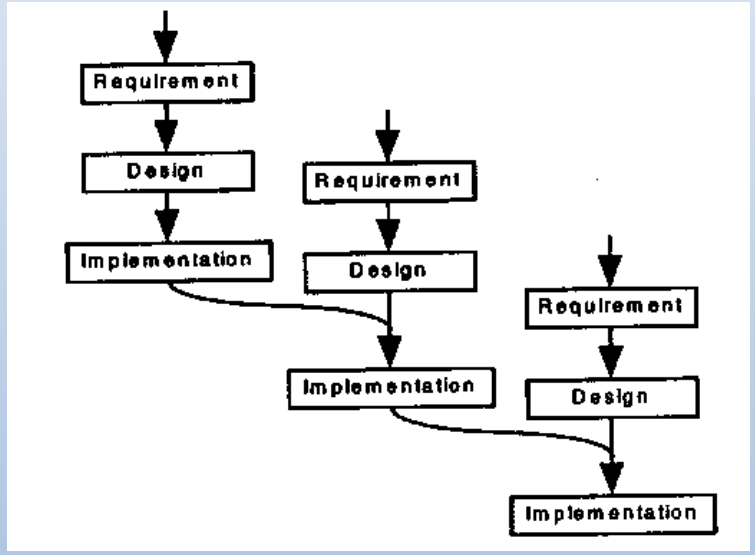
1. Sponsors can view steady progress.
2. This model can be used if the requirements change frequently
3. Communication between the developers and customers can be improved
4. Offers more satisfaction to users.
5. Prototype can be used as a marketing tool.

Prototype Model Disadvantages

1. Difficult to plan the entire project at once, leading to difficulties and inaccuracies with estimating..
2. May encourage an excess of change requests.
3. User can get too involved whereas the program can not be to a high standard
4. Structure of system can be damaged since many changes could be made
5. Not suitable for large applications

Incremental Model

Incremental development is a scheduling and staging strategy
In which the various parts of the system are developed at different times and integrated as they are completed.



Main features of incremental model are

- Increments may be built serially or in parallel
- Each increment adds additional or improved functionality to system
- Requires small group of developers
- Clients can see the system and provide feedback from time to time

Incremental Model Advantages

1. This model gives an opportunity to incorporate user refinements, resulting from experience with earlier releases, into subsequent release
2. At each release an operational product is delivered
3. More flexible - less costly to change scope and requirements.
4. Easier to manage risk because risky pieces are identified and handled during its iteration
5. Generates working software quickly and early during the software life cycle.

Incremental Model Disadvantages

1. Needs good planning, design and careful partitioning of the product

2. Need planned and well – defined interface between increments, especially if they will be developed in parallel.

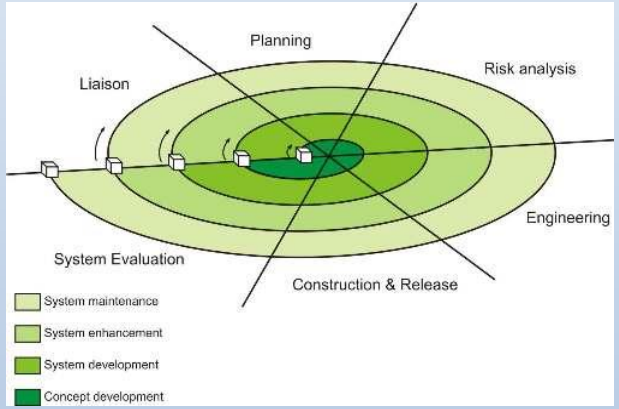
3. Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle

Spiral Model

Spiral Model is combined approach of prototype and waterfall model.

In this model

- Each phase is originated with alternative specifications and risk analysis
- Strengths are evaluated and the necessary amount of testing is proposed for each prototype
- The above process is iterated until customer is satisfied with that prototype
- Final system is constructed based on the refined prototype



Spiral Model Advantages

1. Flexible and easy to accommodate changes
2. New technologies and architectures can be integrated easily
3. Ability to react to risk at each evolutionary level
4. Each iteration of the spiral can be customized to suit the needs of the project
5. The final product or the software is produced early in the software life cycle

Spiral Model Disadvantages

1. More complex planning and management processes
2. Cost may be high
3. It doesn't suit or work well with small projects
4. Highly customized limiting re-usability
5. Risk of not meeting schedule

Rapid Application Development (RAD)

RAD (Rapid Application Development) is a concept with which the products can be developed faster and of higher quality.

The approach focuses on

- Using workshops to gather requirements in fast manner
- Combining the best available techniques in proper sequence to make them effective
- following Prototyping techniques
- Using appropriate tools
- Re-using of software components / modules
- A rigidly paced schedule that defers design improvements to the next product version

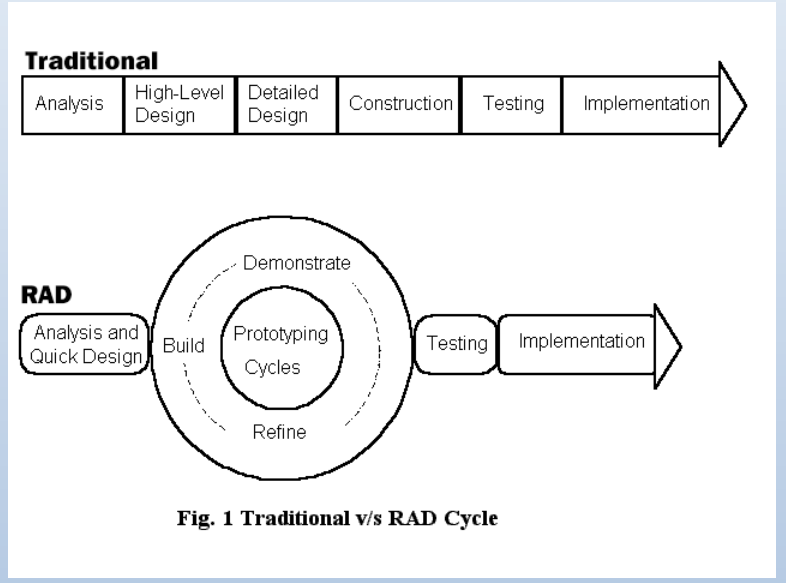


Fig. 1 Traditional v/s RAD Cycle

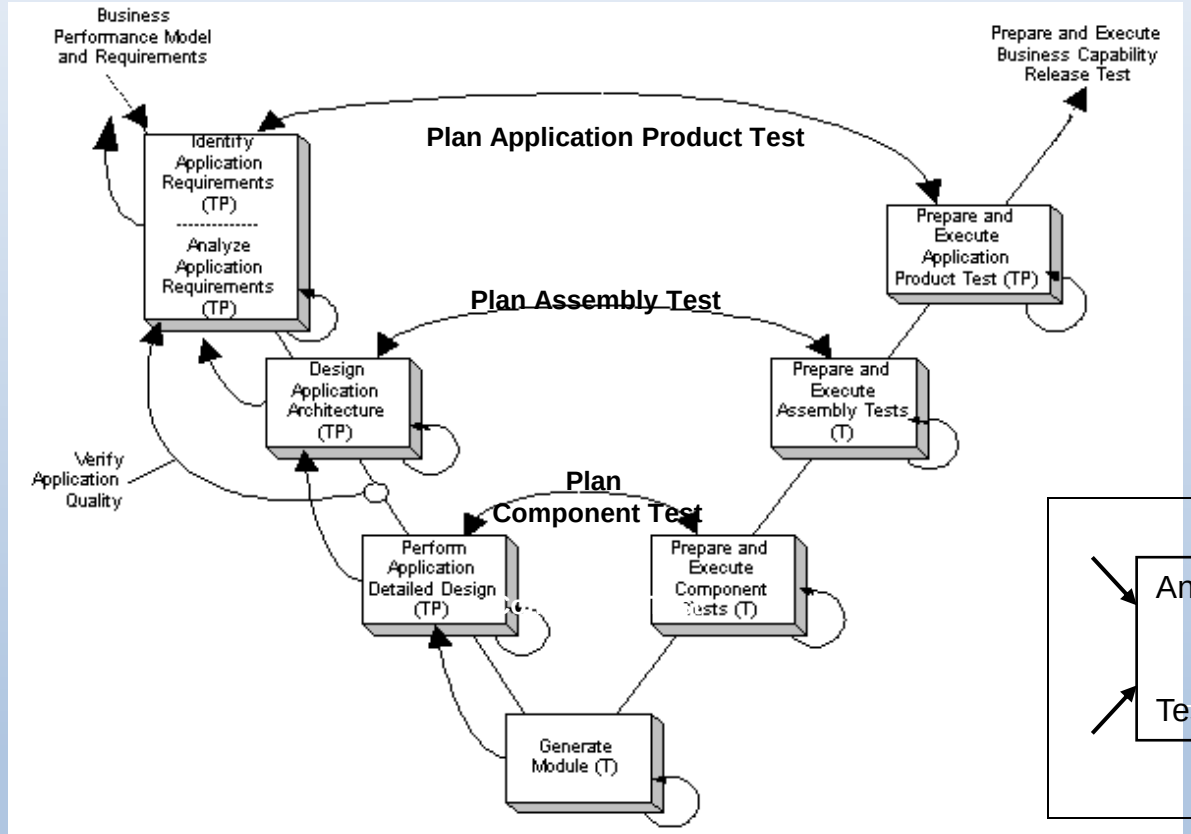
RAD Advantages

1. Cycle time is reduced
2. Customer involvement is high throughout the complete cycle
3. Not achieving customer satisfaction can be minimized
4. Very few resources are needed
5. Modeling concepts are used for capturing the business, data and process

RAD Weaknesses

1. Difficult to implement in legacy systems
2. It requires a system that can be modularized
3. Developers & customers must be prepared for rapid-fire activities in an abbreviated time frame.
4. It may be difficult for many important users to commit the time required for success of the RAD process.
5. This method may not be useful for large, unique or highly complex projects

V - Model

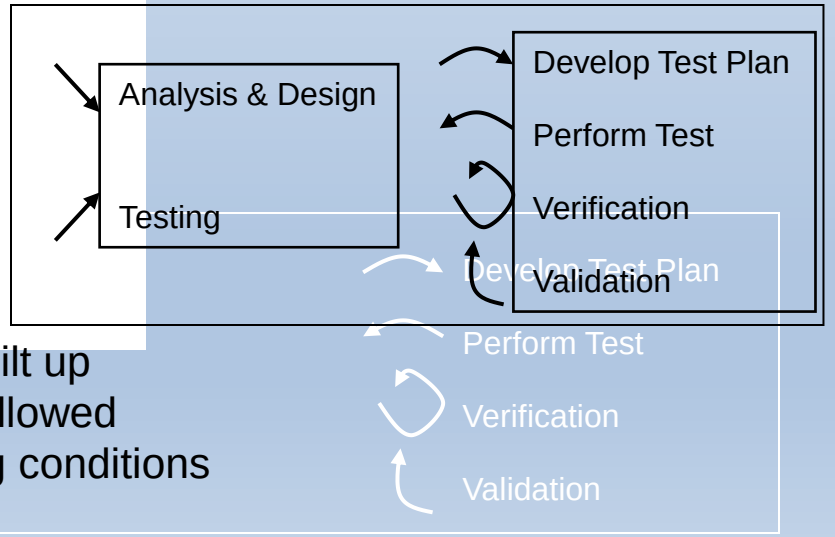


Test Conditions

⬇

Expected Results

- Prevent Overlapping
- Stage Containment
- Root Cause Analysis
  - Errors
  - Defects
  - Faults
- Entry / Exit Criteria



✓ Validation – Checks if the right product is built up
✓ Verification – Checks if the right process is followed
✓ Testing - Checks the product in operating conditions

V-Model Advantages

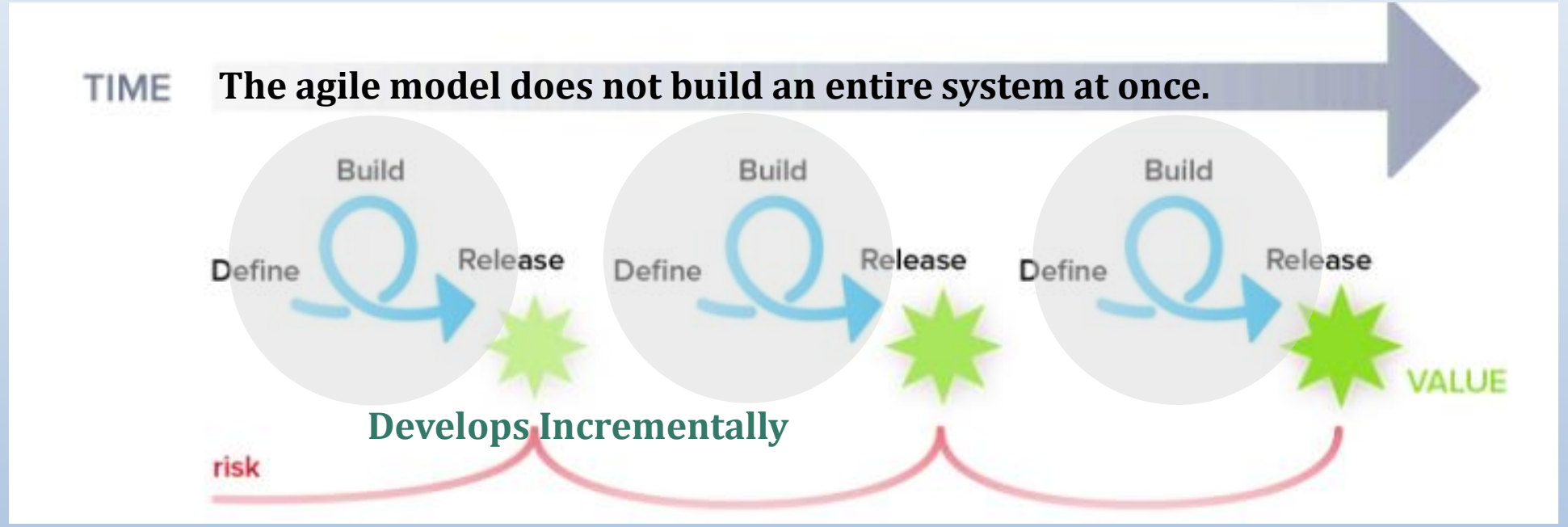
1. Focuses on planning the verification and validation of the product in early stages of product development
2. Rigorous testing efforts are carried out.
3. It is easy to use
4. Progress can be tracked easily.
5. Proactive defect tracking i.e. defects r found at early stages even may be in the development phase before application is tested

V-Model Disadvantages

1. Doesn’t handle iterations or phases
2. It can't easily handle dynamic changes in requirements
3. Require More People to work.

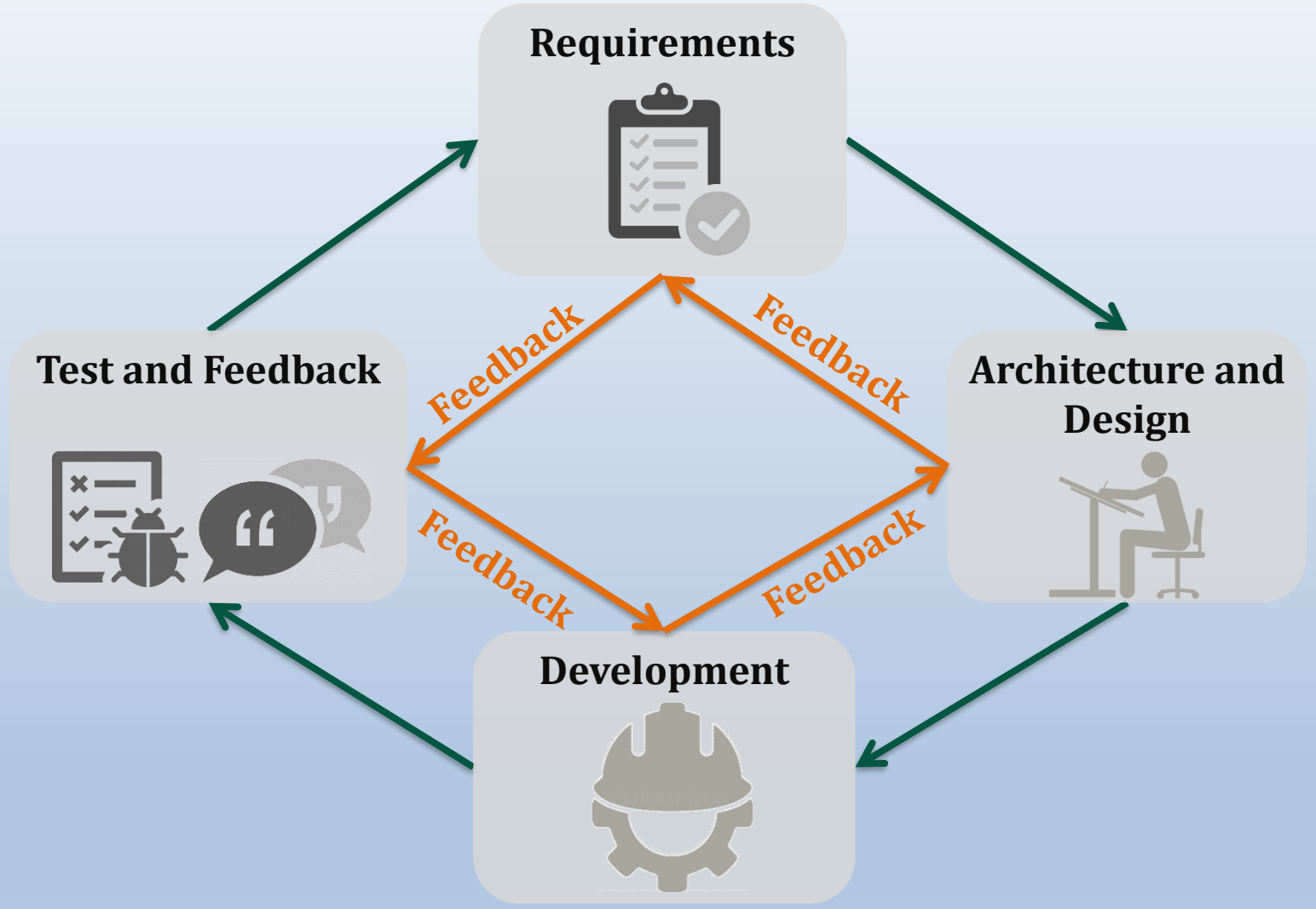
Adaptive or Agile Project Life Cycle

Less time is invested upfront for documenting requirements when development is done incrementally.



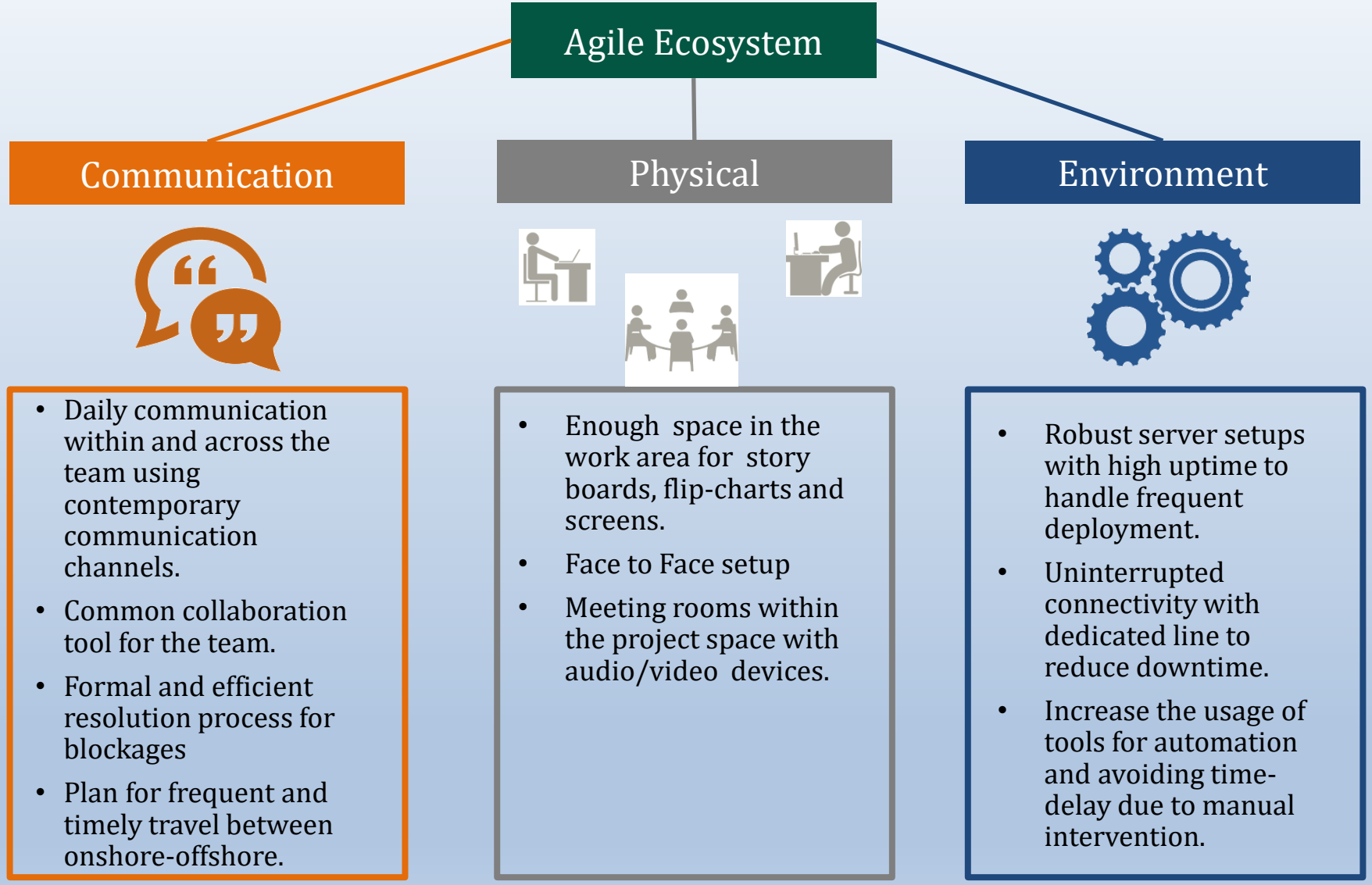
Unlike the more traditional waterfall approach, the agile development method is based on iterative and incremental development.

Adaptive or Agile Project Life Cycle



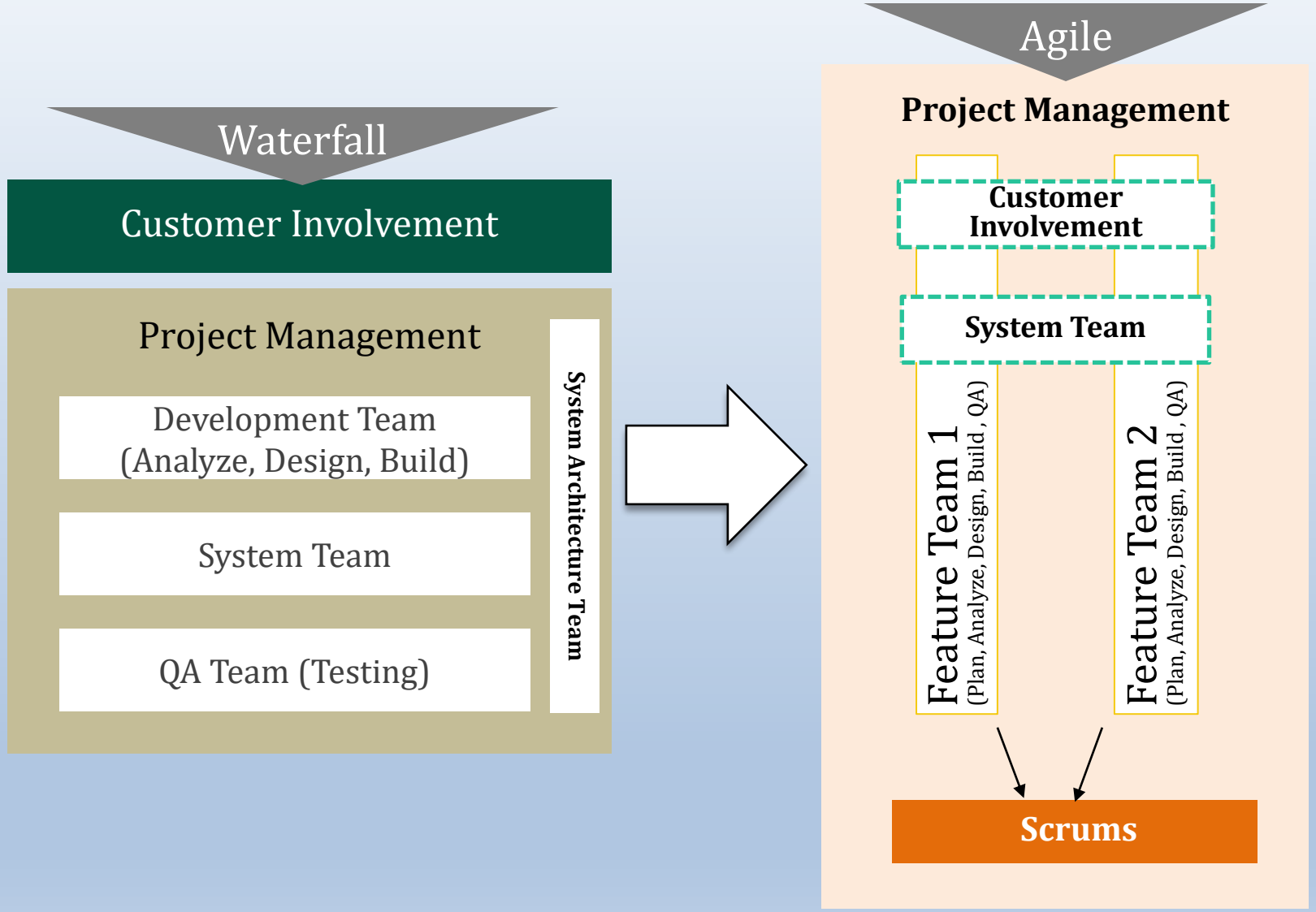
A mainline characteristic of agile software development is that customer feedback occurs simultaneously with development

Agile Ecosystem



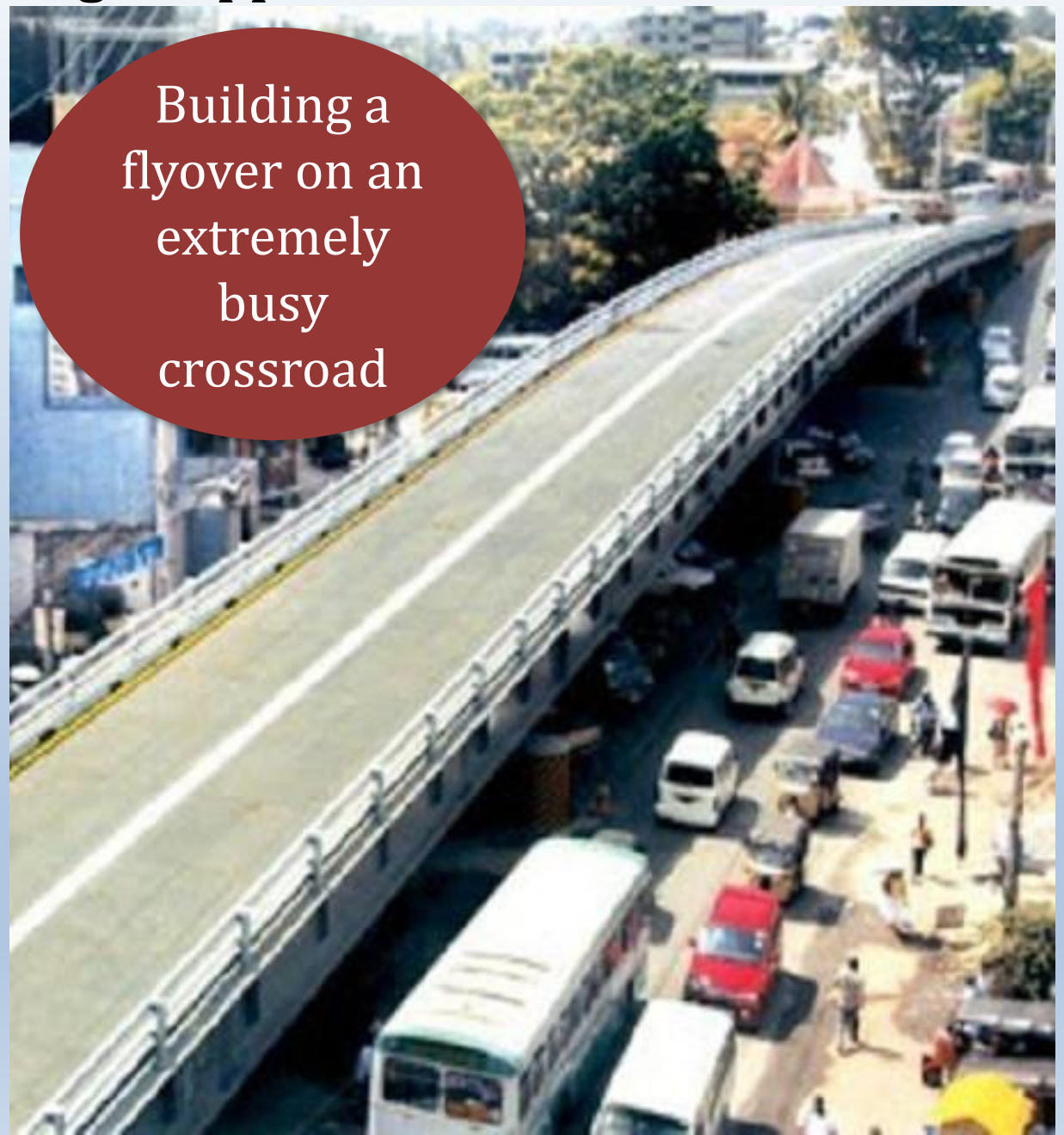
Agility is speed that's what matters when you need to reduce the speed to market!

Team Transformation



Waterfall models gets trans mutated into Agile model as more and more traditional set ups are incorporating some or most of agile within themselves

Example of Agile Approach



- This flyover project demonstrated how incremental delivery can indeed be extremely useful for the project as well as for the end customers.
- The construction was planned to have incremental delivery, so that one direction of the flyover would be constructed before starting the work on the second direction

Example of Agile Approach

01

The overall traffic is still slow, but much better than without any flyovers.

02

Here the end customer (commuter) is using what we call a product of incremental delivery.

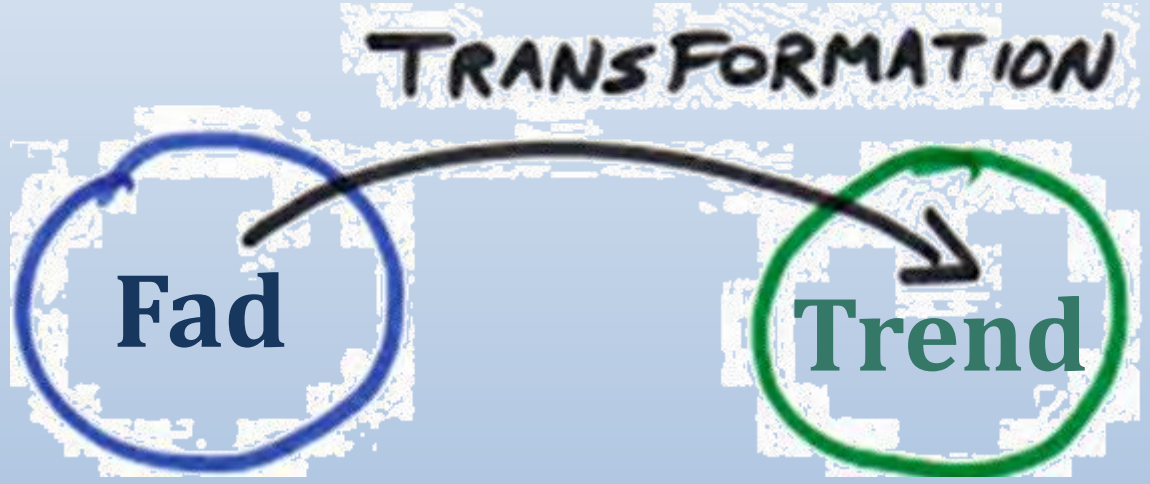
03

This incremental delivery helped customers use the project (the flyover) in nine months instead of waiting twice that long (plus some inevitable delays).

Agile Software Development

Agile is one of the big buzzwords of the IT development industry.

Five years ago,
agile practices transformed from the latest fad to a respectable trend.



As of **2016**, the majority of business analysts we have are experienced or are working in agile teams.

That's because agile is much more widely accepted and adopted now as a discipline.

Agile Software Development

Agile development is a different way of managing IT development teams and projects.

The traditional approach to managing software development projects was failing far too often and there had to be a better way.

The agile manifesto describes 4 important values that are as relevant today as they were then.

"We value individuals and interactions over processes and tools
Working software over comprehensive documentation
customer collaboration over contract negotiation
responding to change over following a plan".

Over the last

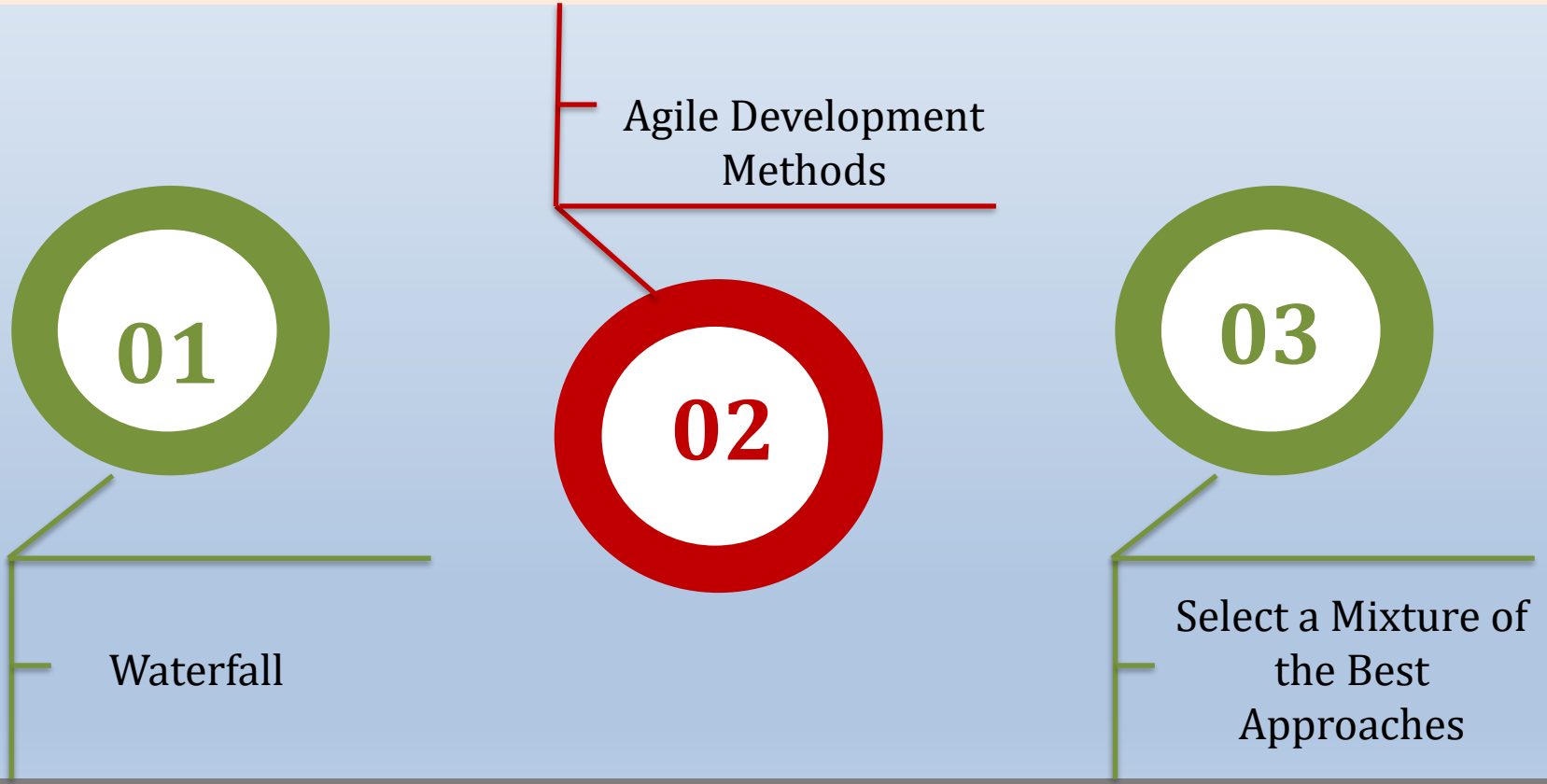
10 years

There is an ever-increasing volume of success stories, where companies have dramatically improved the success and performance of their IT development teams and projects.

Agile Software Development

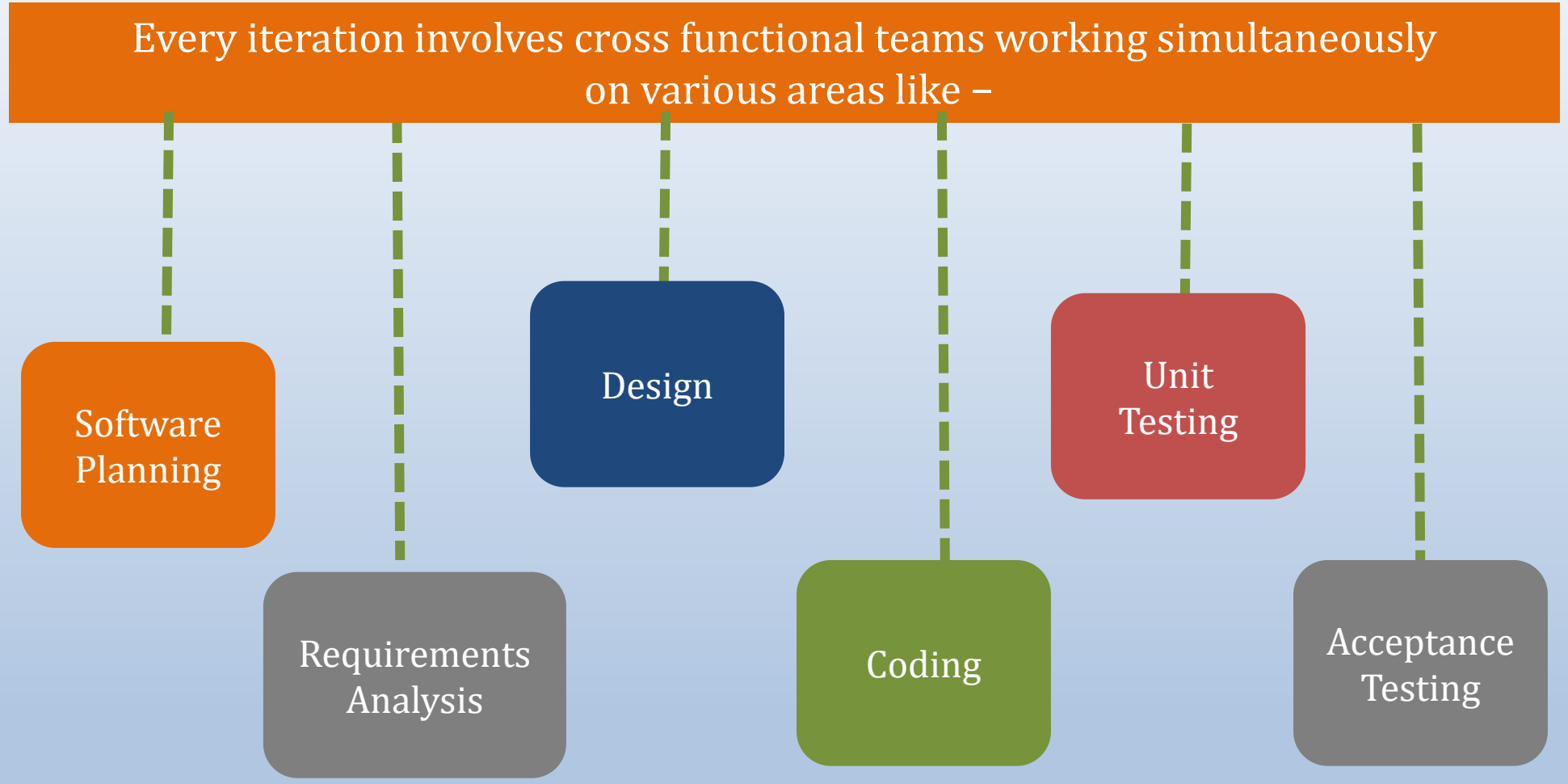
Agile is not a magic bullet for all software development issues.

The real trick is to know lots of techniques from various :



To do this reliably with any degree of success really requires a lot of experience and skill.

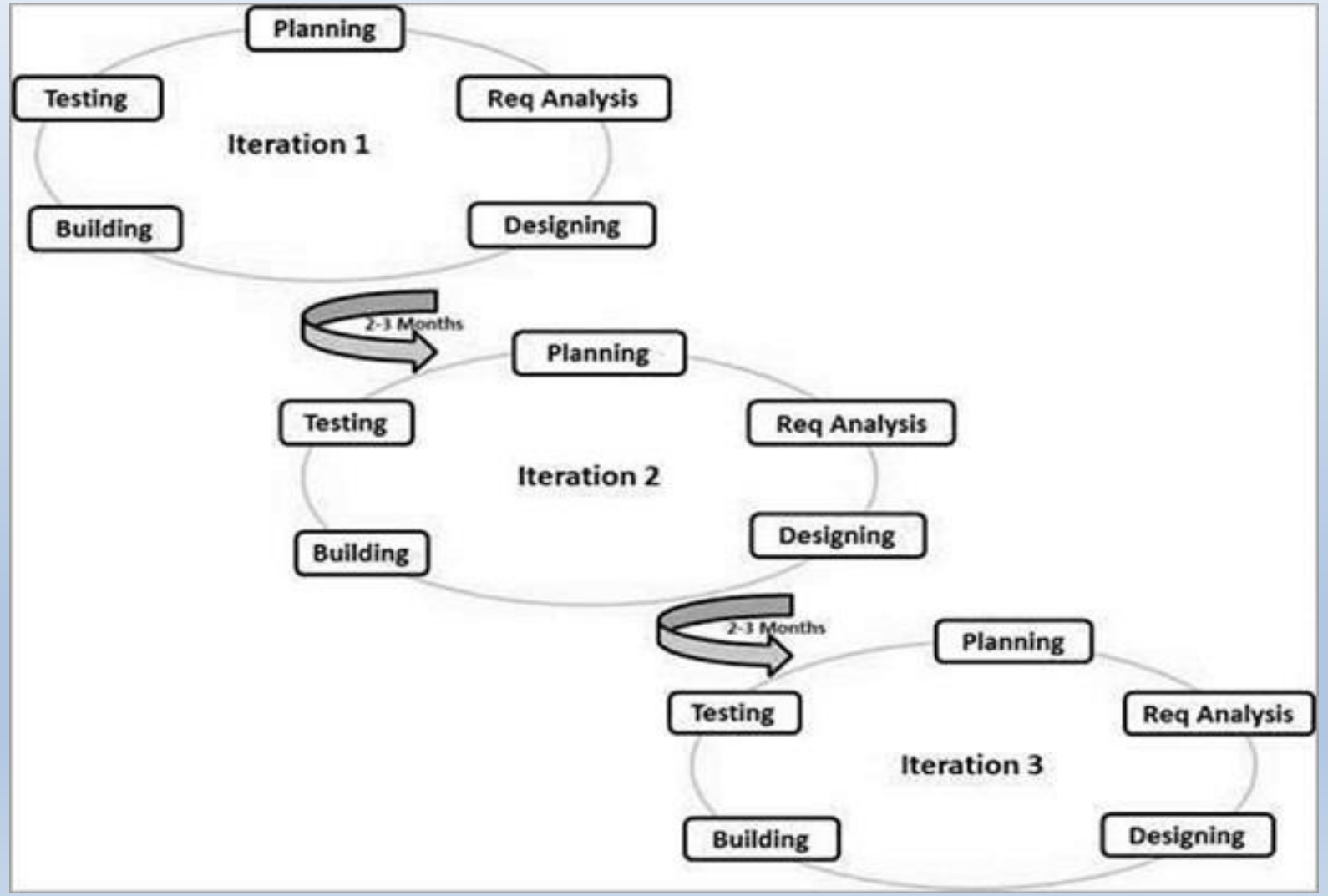
Agile Software Development



At the end of the iteration, a working product is displayed to the customer and important stakeholders.

Agile Software Development

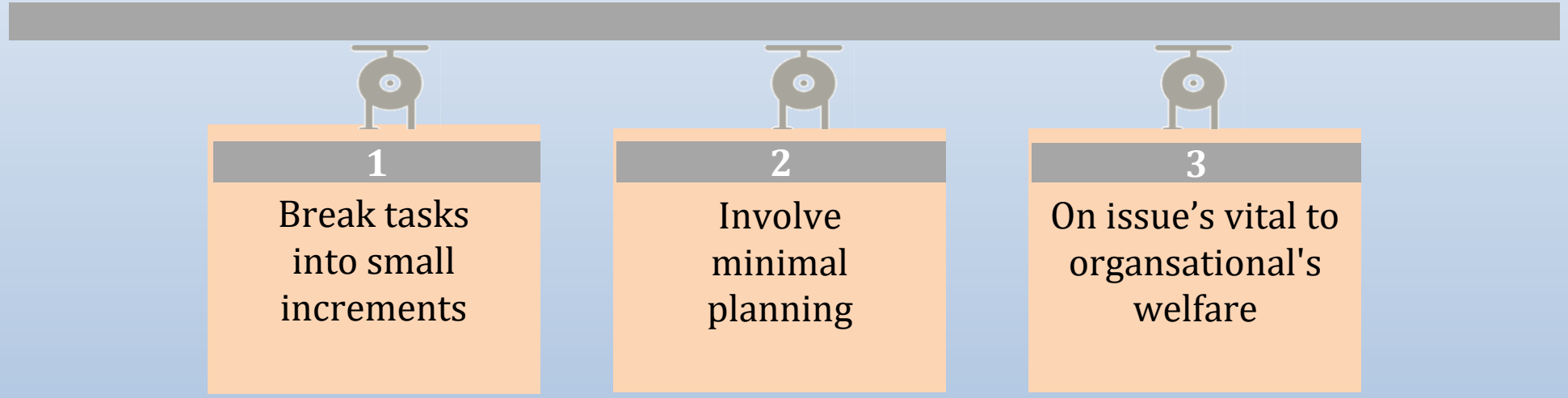
Here is a graphical illustration of the Agile Model –



Agile Software Development

The Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

Agile methods :

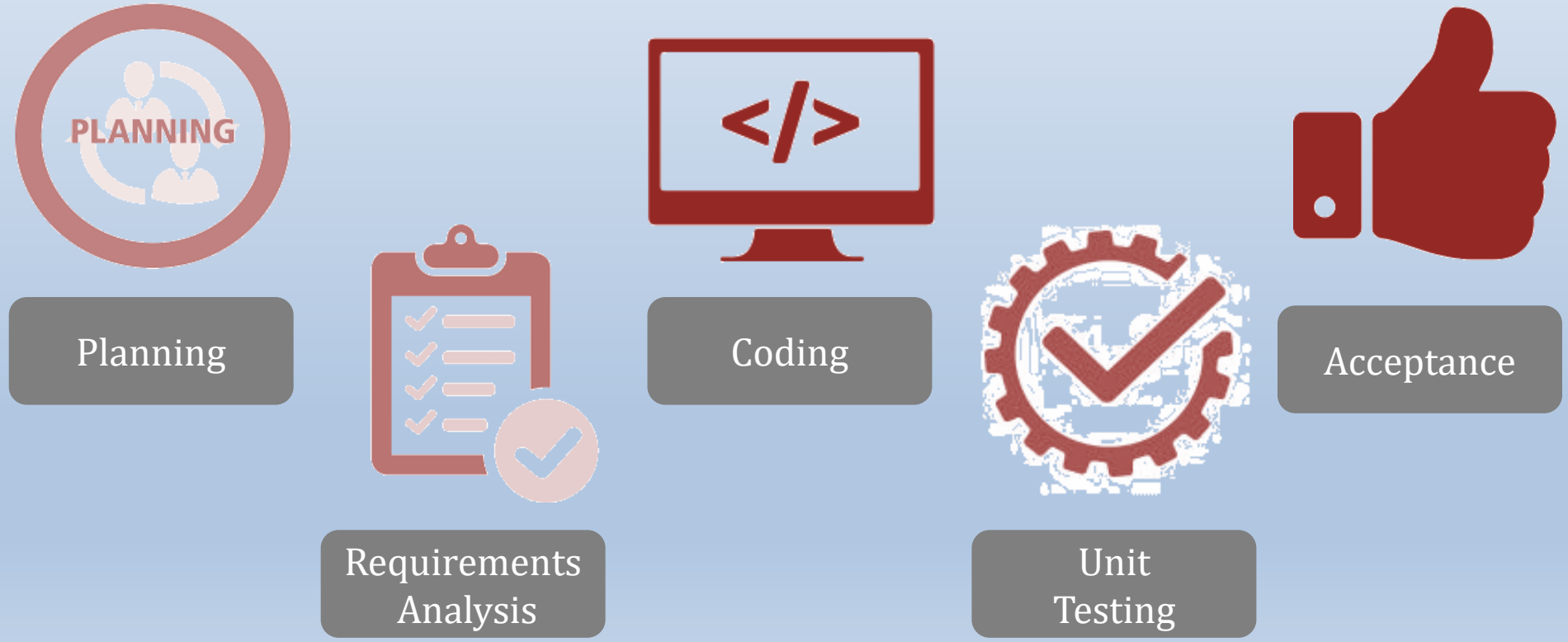


Do not Directly Involve Long-term Planning

Iterations are short time frames that last from one to four weeks.

Agile Software Development

Each iteration involves a team working through a full software development cycle, including:



Agile Software Development

This minimizes overall risk and allows the project to adapt to changes quickly.

An iteration might not add enough functionality to warrant a market release, but the goal is to have an available release (with minimal bugs) at the end of each iteration.

Multiple iterations might be required to release a product or new features.

Agile methods emphasize face-to-face communication over written documents when the team is all in the same location.

Features of Agile

Principle 1: Active user involvement is imperative

Active user involvement is the first principle of agile development.

External users cannot be involved in project development projects

External Customers **Project Development**

In this event it is imperative to have a senior and experienced user representative involved throughout.

Features of Agile

Principle 2: Agile Development Teams Must Be Empowered

An agile development team must include all the necessary team members to make decisions and make them on a timely basis.

The team must establish and clarify and prioritise requirements, agree to the tasks required to deliver, and estimate the effort involved.

Features of Agile

Principle 3: Requirements evolve, but timescales are fixed.

01	Agile development works on the premise that requirements emerge and evolve, and that however much you analyze and design, this will always be the case because you cannot really know for sure what you want until you see and use the software.
02	In the time, you would have spent analyzing and reviewing requirements and designing a solution, external conditions could also have changed.
03	Agile development projects accept change; in fact, they expect it.
04	In these projects, requirements are allowed to evolve, but the timescale is fixed.

Features of Agile

Principle 3: Requirements evolve, but timescales are fixed.

05	To include a new requirement, or to change a requirement, the user or product owner must remove a comparable amount of work from the project in order to accommodate the change.
06	This ensures the team remains focused on the agreed timescale and allows the product to evolve into the right solution.
07	It does, however, also pre-suppose that there's enough non-mandatory features included in the original timeframes to allow these trade-off decisions to occur without fundamentally compromising the end product.

Features of Agile

Principle 4: Agile Requirements are barely sufficient

Agile development teams capture requirements at a high level and on a piecemeal basis, just-in-time for each feature to be developed.

Agile requirements are ideally visual and should be barely sufficient, i.e. the absolute minimum required to enable development and testing to proceed with reasonable efficiency.

The rationale for this is to minimise the time spent on anything that doesn't actually form part of the end product.

Features of Agile

Principle 5: Done means done!

01 Features developed within iteration i.e. a sprint in scrum, should be 100% complete by the end of the sprint.

02 Too often in software development, “done” doesn’t really mean “done!”, tested, styled and accepted by the product owner. It just means developed.

03 Make sure that each feature is fully developed, tested, styled, and accepted by the product owner before counting it as “DONE!”.

04 If there is any doubt about what activities should or shouldn’t be completed within the sprint for each feature, “DONE!” should mean shippable.

Features of Agile

05 Multiple features can be developed in parallel in a team situation.

06 However, within the work of each developer, do not move on to a new feature until the last one is shippable.

07 This is important to ensure the overall product is in a shippable state at the end of the sprint, not in a state where multiple features are 90% complete or untested, as is more usual in traditional development projects.

Features of Agile

Principle 6: Agile testing is not for dummies!

Testing is integrated throughout the software development lifecycle.

Agile development does not have a separate test phase as such.

Developers are much more heavily engaged in testing, writing automated repeatable unit tests to validate their code.

Features of Agile

Principle 6: Agile testing is not for dummies!

1. With automated repeatable unit tests, testing can be done as part of the build, ensuring that all features are working correctly each time the build is produced.
2. And builds should be regular, at least daily, so integration is done as you go too.
3. The purpose of these principles is to keep the software in releasable condition throughout the development, so it can be shipped whenever it's appropriate.

Scrum

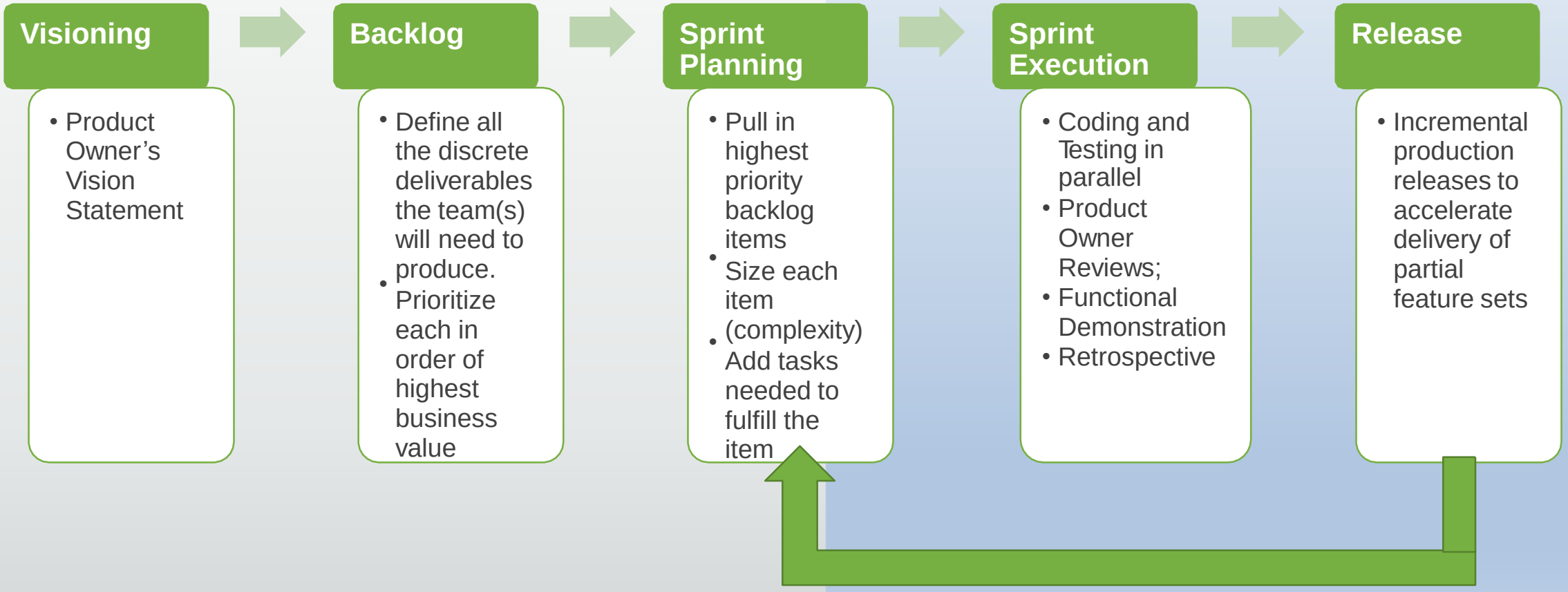
Overview of the Scrum Practice Framework

Sujata Batra

Scrum

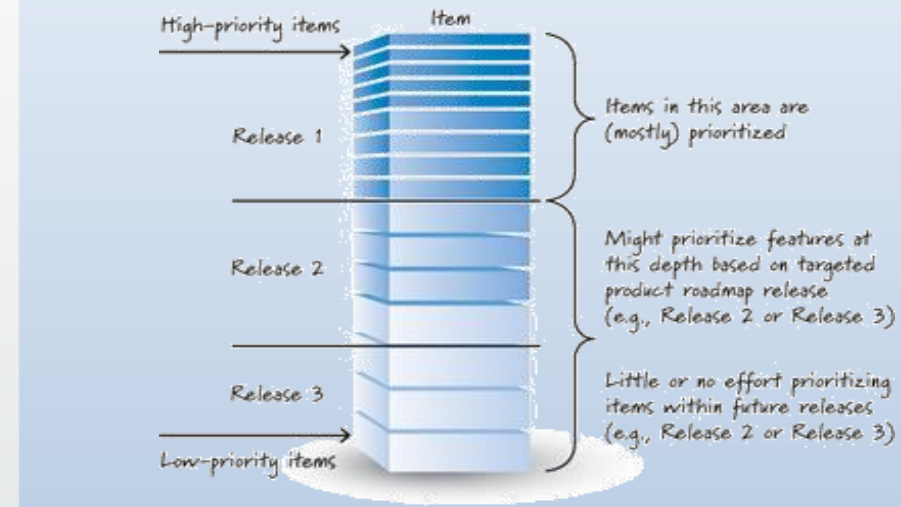
Scrum is the framework that helps teams work together.
Gets its name from a Rugby term used as a metaphor to reflect the degree of team cooperation needed to advance the football across the goal line.

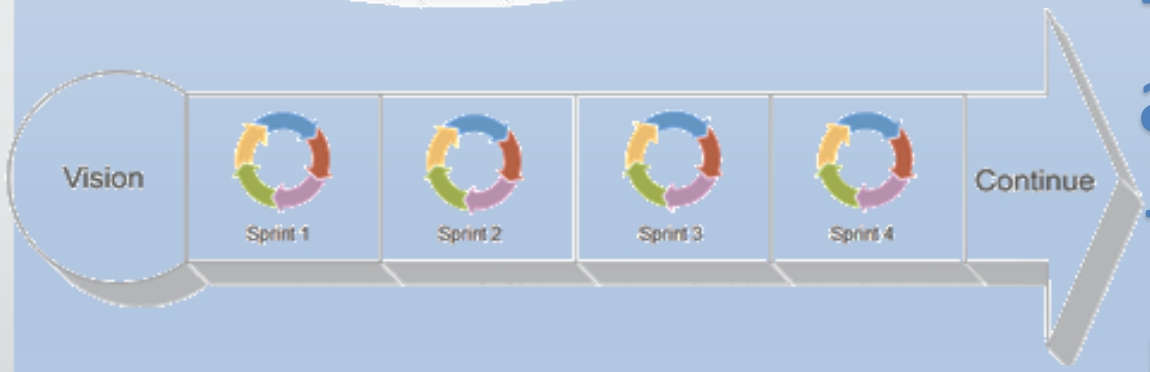
How Does It Work



How Does It Work

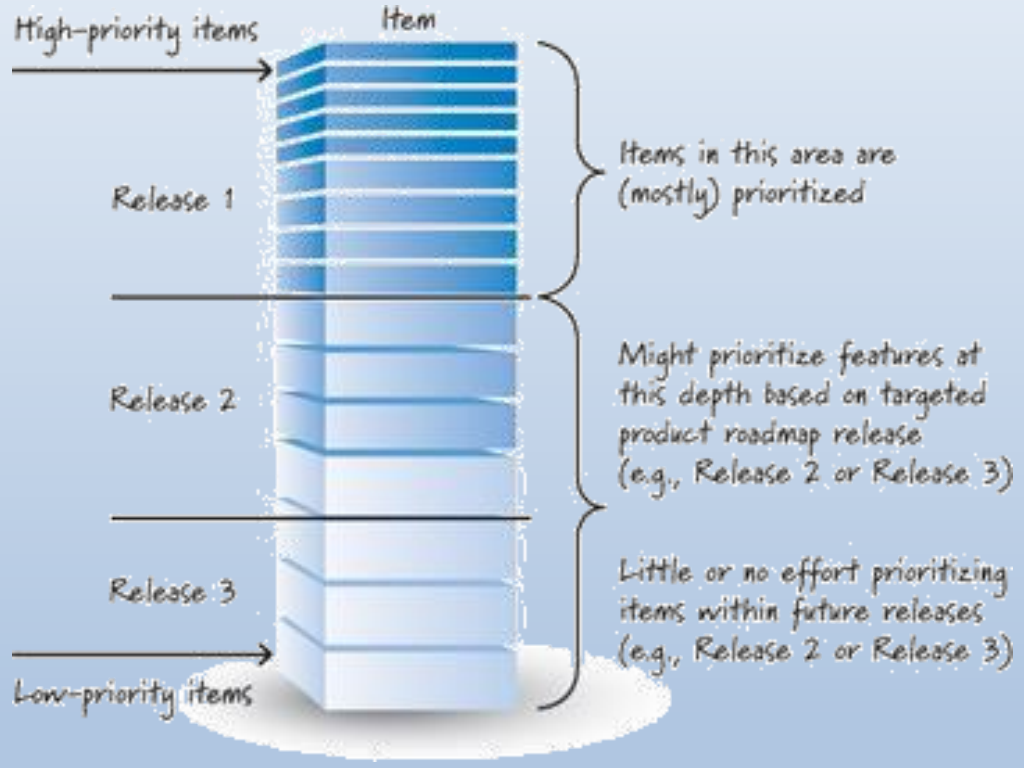
- Product Backlog: Single Source of Functional and non-functional requirements.
- Chops up the Product Backlog into a series of smaller pieces
- Each piece is worked within a time boxed period called a Sprint.
- Work is inspected, accepted or rejected each Sprint by the Product Owner (business owner).





How Does It Work

- Business Value
  - Work is prioritized highest business value to lower business value.
  - Highest value items should be elaborated in detail; ready for the next Sprint Planning.
- Tactics
  - MoSCoW (must have, should have, could have, won't have)
  - REIO (Required, Essential, Important, Optional)
  - Cost – Benefit Matrix



User Story

Describes a small discrete "need" from the perspective of a role or persona.

As a [call center agent] (WHO)

I need to [login with my password] (WHAT)

So that [I can access the customer's reservation to cancel it] (WHY)

Contains acceptance criteria that defines "done" (story is done when . . .)

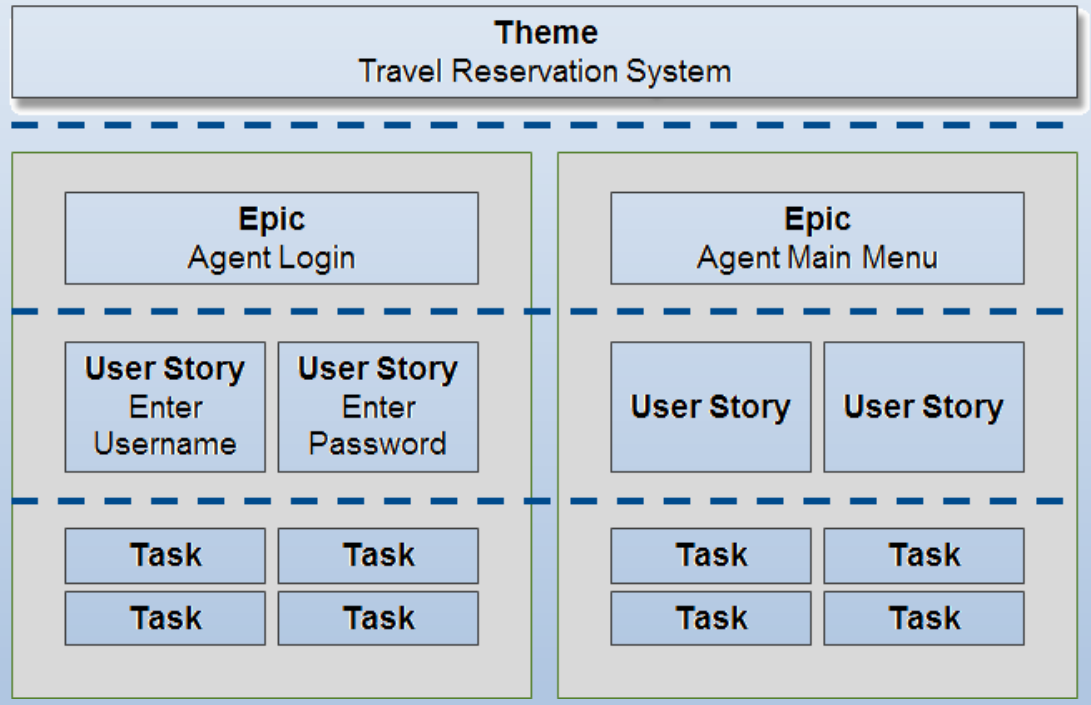
- ❑ a premium member can cancel same day without a fee
- ❑ a non-premium member is charged 10% for a same-day cancellation
- ❑ email confirmation is sent to the customer
- ❑ hotel is notified of the cancellation

User Story

- Contains tasks that describe the actions and estimated effort required to fulfill the Story need.
  - Typically starts with a verb, concise, and self evident what the action is and an estimate of effort
    - Create User Table (1 hr)
    - Create password encryption service (4 hr)
    - Create login service (4 hr)
- Is testable (functionally)
  - Well constructed acceptance criteria doubles as functional test criteria for the story (positive and negative)
    - User can login using a valid password
    - User cannot login using an invalid password

User Story Scope

- Theme
  - Very broad high level category of related Epics and Stories
- Epic
  - High level User Story; typically representing a broad functional feature
  - Epics are sometimes referred to as Feature
- User Story
  - Represents a discreet detailed functional requirement.



Story Map

- Make visible the workflow or value chain
- Show relationships of larger stories to child stories
- Help confirm the completeness of the Backlog
- Provide a useful context for prioritization
- Plan releases in complete slices of functionality

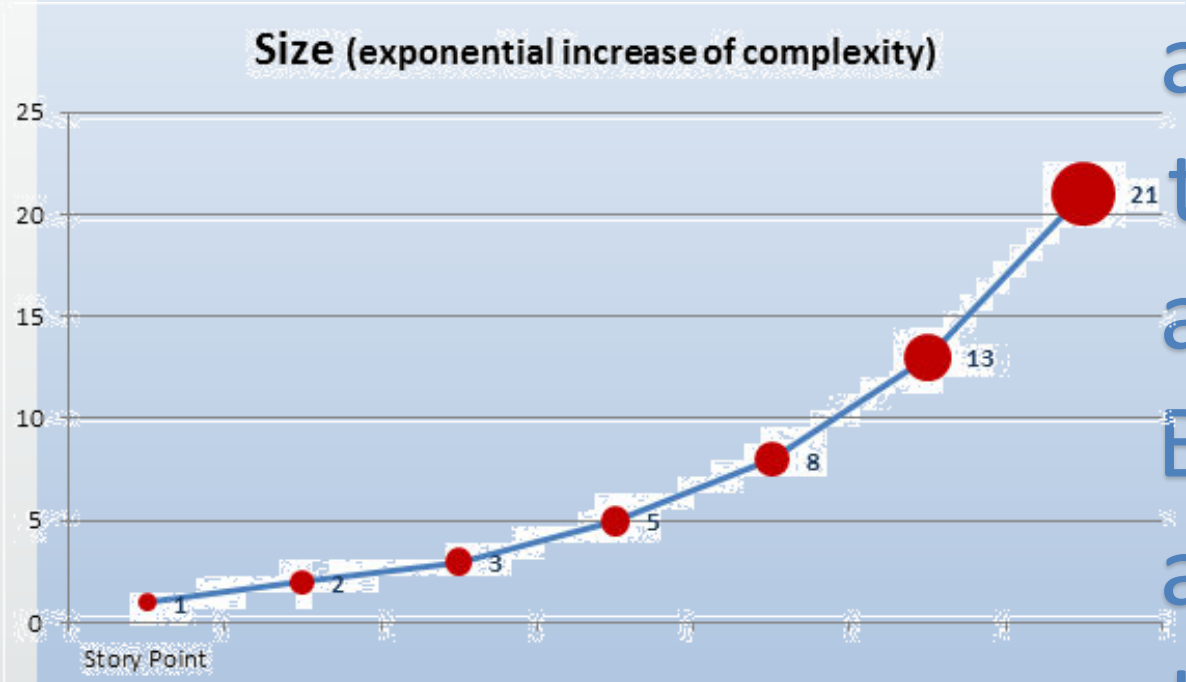
Release Roadmap

- Helps align stakeholder expectations
- List the Release Name or Version Number
- List the goals for each release
- List the Preliminary feature set for each release
- Optionally include metrics that help define if the goal(s) were met

Timeline	2016Q1	2016Q2	2016Q3	2016Q4
Rel ID	R1	R2	R3	R4
Goal	One UI; all admin systems, basic search functions	Add additional search types	Integration of IVR pop, SWAP, and CLASS	Additional Notes Functionality
Features	• Name search • Organization search • Policy number search • View Contract details (Summary) • Search usage reporting	• Customer search using last 4 of SSN • Search using FULL SSN • Adjustments to Agent Result Data • Search usage reporting adjustments	• IVR Pop integration • View note by Policy Number and Owner • SWAP Integration CLASS Integration	• Attention and Alert note handling • Copy/paste functionality • Ability to enter notes on UI and write back to source system

Estimating

- Story Points
  - Variation of tee-shirt sizing estimated in points relative to perceived complexity of the story (effort, complexity, and risk)
  - Much quicker and accurate than time spent 'breaking down and measuring'
- Techniques
  - Planning poker cards
  - Reference Story. 2 story points = 'small', size other User Stories relative to that; smaller, larger, same.



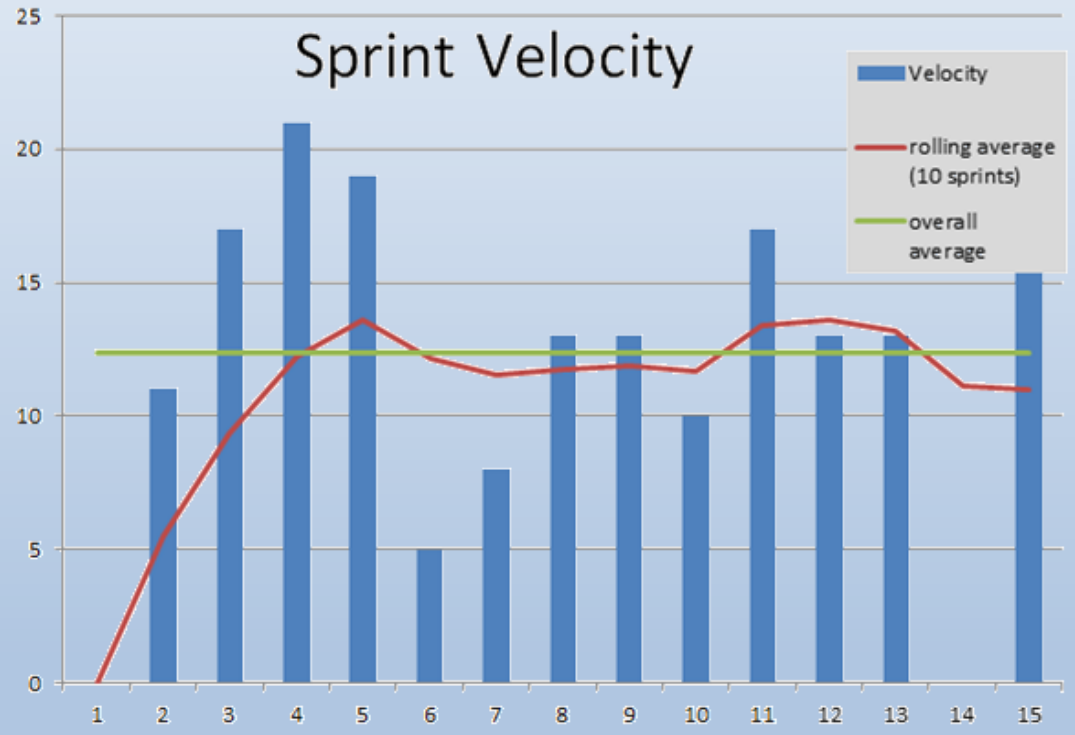
Relative Estimation Advantage

- Humans are terrible at absolute estimation but quite good at relative estimation.
- It is generally faster
- It gets a team thinking (and talking) as a group, rather than as individuals
- It encourages spending analysis time appropriately
- It is cost-effective

Animal	Estimate the weight in pounds	Estimate the weight lightest (1) to Heaviest (5)
Tiger	?	4
Rabbit	?	2
Squirrel	?	1
Elephant	?	5
Impala	?	3

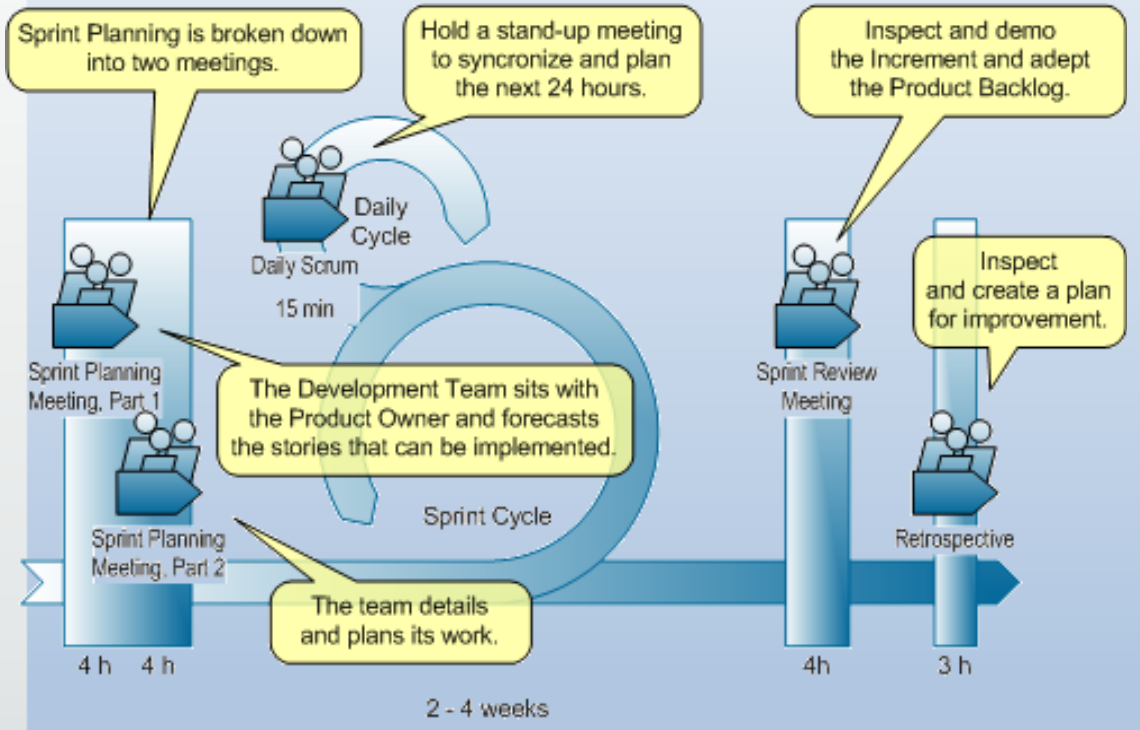
Velocity

- Points total from all completed stories is the team's velocity for that Sprint.
- After several Sprints, velocity "norms". Average velocity then becomes a predictor of Sprint throughput.
- The team can periodically compute estimated project completion based on backlog remaining points



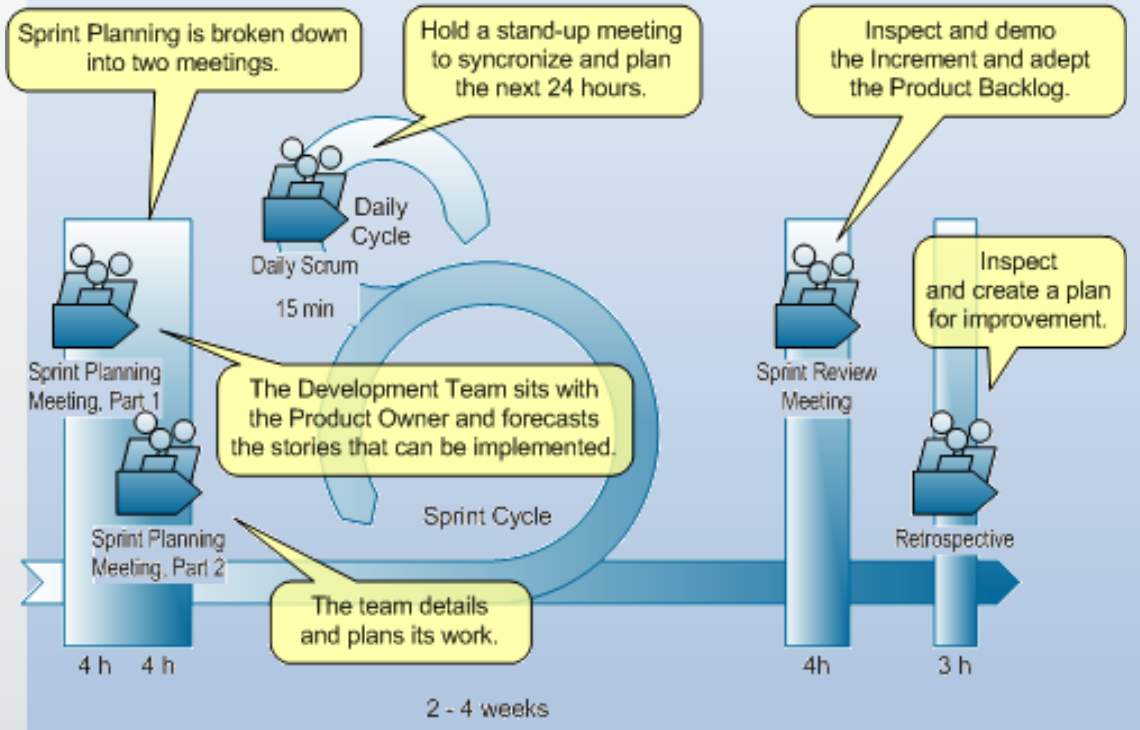
Sprints

- Time boxed
  - Typically 2-4 weeks
- Sprint Planning (Day 1)
  - Pull in the next highest priority items from the backlog.
  - First session with the Product Owner
  - Second session to work out the technical strategy for completing the work.



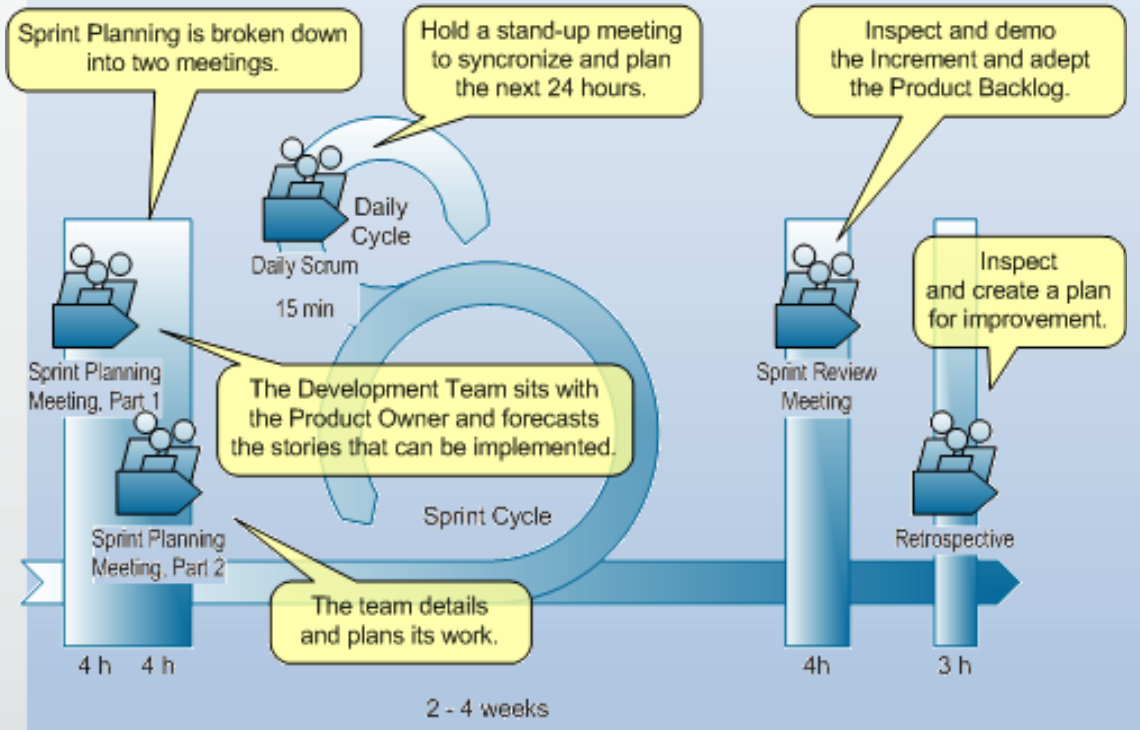
Sprints

- Daily Stand-up (Each Day)
  - Each team member:
    - What they did yesterday
    - What they plan to do today
    - Any impediments blocking progress.
- Sprint Review (Final Day)
  - Product Owner reviews achievements of the Sprint with the team
  - Product Owner and team presents a demonstration or discusses latest functionality with external audience.



How Does It Work

- Retrospective
  - The Retrospective, or ‘Retro’, is attended by the Scrum Master and the team and is the final team meeting in the Sprint.
  - The primary purpose is to determine what went well, what didn't go well, and how the team can improve in the next Sprint.
  - The Retrospective is the opportunity for the team to focus on its overall performance and identify strategies for continuous improvement on its processes



Roles

Product Owner

- Represents the Business
- Defines requirements (the backlog)
- Accepts or rejects team output
- Makes business decisions
- Provides visibility to leadership

Scrum Master

- Scrum process expert
- Ensures consistent team practices
- Coaches team and individuals; to maximize efficiency and quality
- Partners with the Product Owner to maximize alignment
- Assists with logistics, admin, or impediment removal to ensure team can run full throttle.

IT Team

- Typically 7 +/- 2 members
- Armed with skills to deliver increments of working software
- The team is empowered to organize/execute work and to solve problems within their control
- Cross-functional; members learn a bit of how other work is done so they can assist as needed.

Scrum Master

- Duties and Allocations
  - People: Gate keeper; shield the team from undue interruptions and distractions, build and maintain communication between the team and everybody else external to the team.
  - Process: Scrum process activities and meetings.
  - Delivery: Ongoing backlog refinement sessions, impediment management, delivery coordination and status meetings, governance / PMO administrative tasks.

Scrum Master Duties and Time Allocations (approximate)	**2 Week (10 day) Sprint**		**3 Week (15 day) Sprint**	
Gross Capacity	80	Hours	120	Hours
People	10	13%	15	13%
Gatekeeper: Interface point between team and management or stakeholders. Shield the team from undue interruptions.	10		15	
Relationships management; help build and maintain communication and trust within the team and between the team and everybody else external to the team.				
Process	19	24%	22	18%
Daily SCRUM Meetings	5		8	
Sprint Planning Meeting	8		8	
Sprint Review Meeting	3		3	
Sprint Retrospective Meetings	3		3	
Delivery	36	45%	54	45%
Ongoing Backlog Refinement	12		18	
Impediment Management	10		15	
Delivery coordination and status meetings	10		15	
Governance / PMO administrative tasks	4		6	
Uncommitted Hours	15	19%	29	24%
Utilization	**81%**		**76%**	

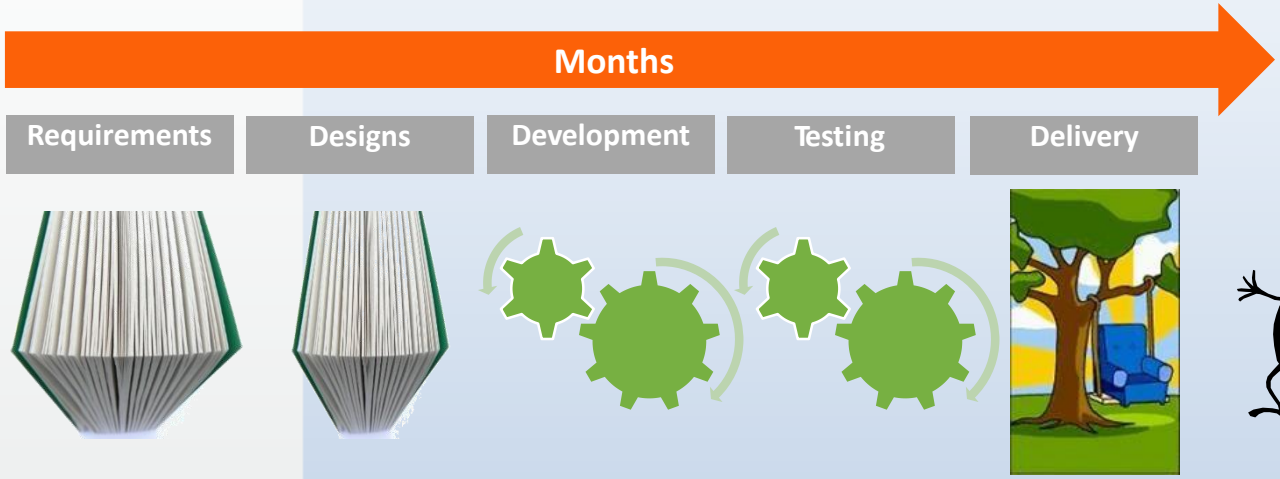
Business Analyst

- Assists the Product Owner and the Team
  - The Product Owner has a full time job
  - The Product Owner defines the high level functional deliverables (Epics) and priority
  - The BA digs out the detail of each high level functional deliverable into users stories
  - The BA helps create minimum needed designs
  - Pre-Validates the Story as “Done”
  - Helps prepare and execute test plans

Putting It All Together

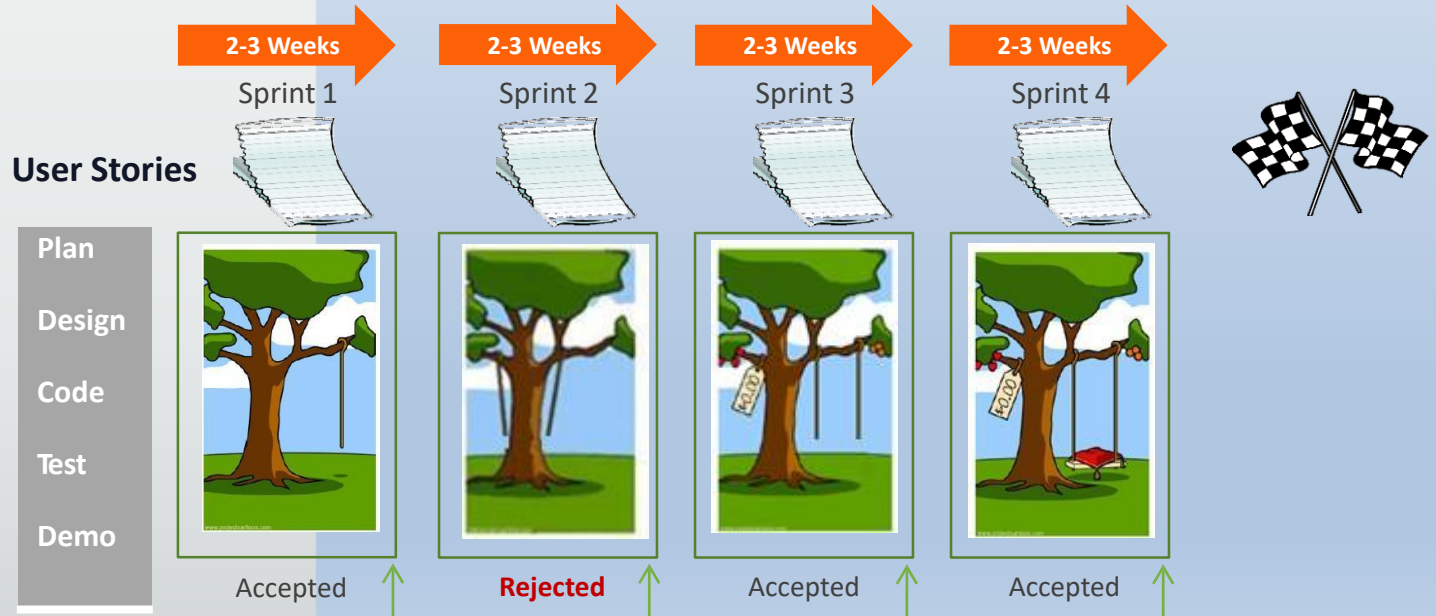
The Customer Needs This

Waterfall



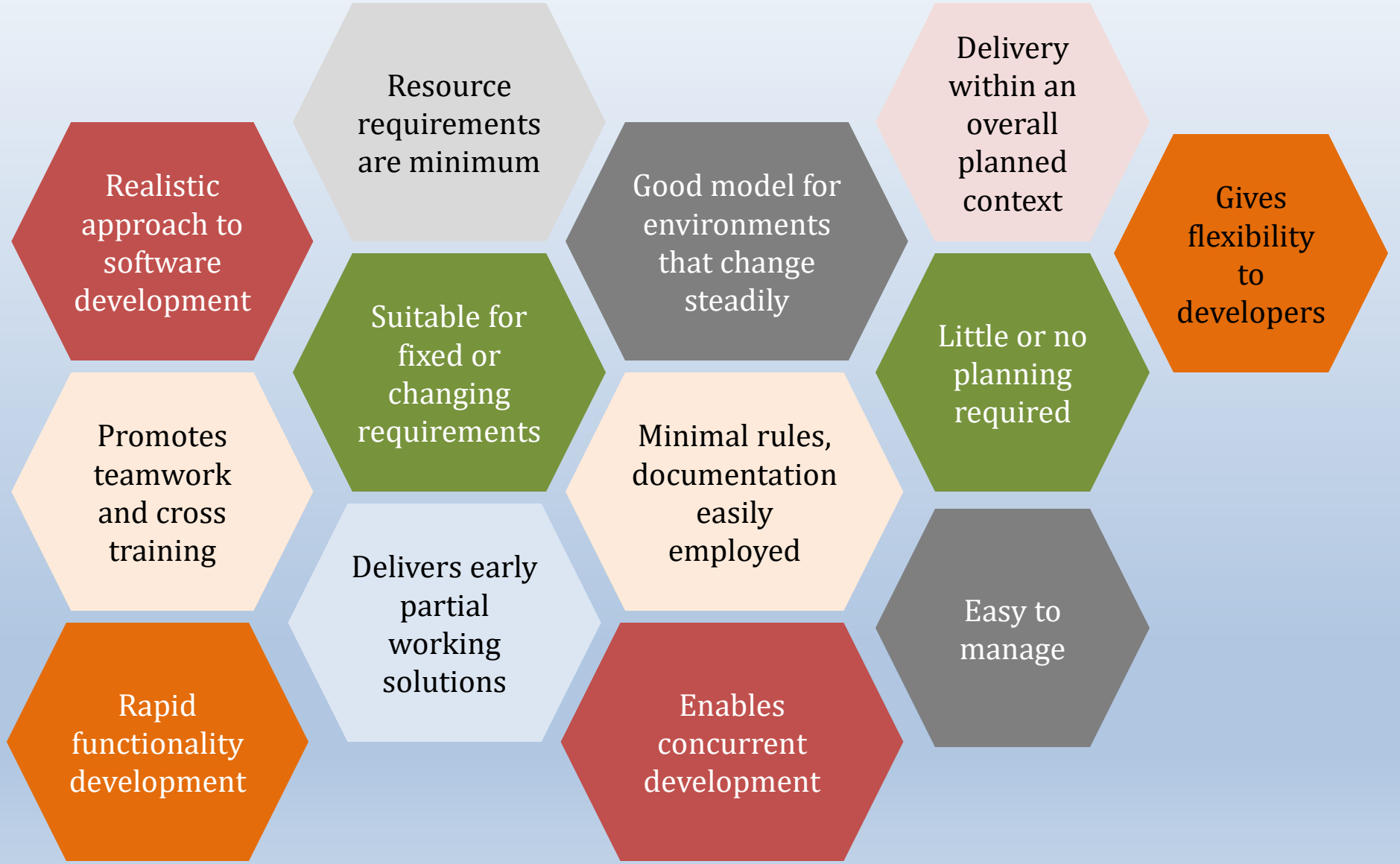
Agile (Scrum)

1 User Story = 1 Functional Requirement



- Defects discovered and corrected within each Sprint
- Product Owner / Customer sees functionality each Sprint, accepts or rejects

Agile Model - Advantages



Agile Model - Disadvantages

1. Not suitable for handling complex dependencies.
2. More risk of sustainability, maintainability and extensibility.
3. Overall plan is a must.
4. Strict delivery management to meet deadlines.
5. Depends heavily on customer interaction.
6. Very high individual dependency.
7. Technology transfer is challenging.
8. Lack of documentation.